

Frontiers
in
Artificial
Intelligence
and
Applications

INTELLIGENT TECHNOLOGIES – THEORY AND APPLICATIONS

NEW TRENDS IN INTELLIGENT TECHNOLOGIES

Edited by
Peter Šinčák
Ján Vaščák
Vladimír Kvasnička
Jiří Pospíchal

IOS
Press
OHM
Ohmsha

VISIT...

LANZAROTE
Caliente.COM

INTELLIGENT TECHNOLOGIES – THEORY AND APPLICATIONS

Frontiers in Artificial Intelligence and Applications

*Series Editors: J. Breuker, R. López de Mántaras, M. Mohammadian, S. Ohsuga and
W. Swartout*

Volume 76

Previously published in this series:

- Vol. 75, I. Cruz et al. (Eds.), *The Emerging Semantic Web*
- Vol. 74, M. Blay-Fornarino et al. (Eds.), *Cooperative Systems Design*
- Vol. 73, In production
- Vol. 72, A. Namatame et al. (Eds.), *Agent-Based Approaches in Economic and Social Complex Systems*
- Vol. 71, J.M. Abe and J.I. da Silva Filho (Eds.), *Logic, Artificial Intelligence and Robotics*
- Vol. 70, B. Verheij et al. (Eds.), *Legal Knowledge and Information Systems*
- Vol. 69, N. Baba et al. (Eds.), *Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies*
- Vol. 68, J.D. Moore et al. (Eds.), *Artificial Intelligence in Education*
- Vol. 67, H. Jaakkola et al. (Eds.), *Information Modelling and Knowledge Bases XII*
- Vol. 66, H.H. Lund et al. (Eds.), *Seventh Scandinavian Conference on Artificial Intelligence*
- Vol. 65, In production
- Vol. 64, J. Breuker et al. (Eds.), *Legal Knowledge and Information Systems*
- Vol. 63, I. Gent et al. (Eds.), *SAT2000*
- Vol. 62, T. Hruška and M. Hashimoto (Eds.), *Knowledge-Based Software Engineering*
- Vol. 61, E. Kawaguchi et al. (Eds.), *Information Modelling and Knowledge Bases XI*
- Vol. 60, P. Hoffman and D. Lemke (Eds.), *Teaching and Learning in a Network World*
- Vol. 59, M. Mohammadian (Ed.), *Advances in Intelligent Systems: Theory and Applications*
- Vol. 58, R. Dieng et al. (Eds.), *Designing Cooperative Systems*
- Vol. 57, M. Mohammadian (Ed.), *New Frontiers in Computational Intelligence and its Applications*
- Vol. 56, M.I. Torres and A. Sanfeliu (Eds.), *Pattern Recognition and Applications*
- Vol. 55, G. Cumming et al. (Eds.), *Advanced Research in Computers and Communications in Education*
- Vol. 54, W. Horn (Ed.), *ECAI 2000*
- Vol. 53, E. Motta, *Reusable Components for Knowledge Modelling*
- Vol. 52, In production
- Vol. 51, H. Jaakkola et al. (Eds.), *Information Modelling and Knowledge Bases X*
- Vol. 50, S.P. Lajoie and M. Vivet (Eds.), *Artificial Intelligence in Education*
- Vol. 49, P. McNamara and H. Prakken (Eds.), *Norms, Logics and Information Systems*
- Vol. 48, P. Návrat and H. Ueno (Eds.), *Knowledge-Based Software Engineering*
- Vol. 47, M.T. Escrig and F. Toledo, *Qualitative Spatial Reasoning: Theory and Practice*
- Vol. 46, N. Guarino (Ed.), *Formal Ontology in Information Systems*
- Vol. 45, P.-J. Charrel et al. (Eds.), *Information Modelling and Knowledge Bases IX*
- Vol. 44, K. de Koning, *Model-Based Reasoning about Learner Behaviour*
- Vol. 43, M. Gams et al. (Eds.), *Mind Versus Computer*
- Vol. 41, F.C. Morabito (Ed.), *Advances in Intelligent Systems*
- Vol. 40, G. Grahne (Ed.), *Sixth Scandinavian Conference on Artificial Intelligence*
- Vol. 39, B. du Boulay and R. Mizoguchi (Eds.), *Artificial Intelligence in Education*
- Vol. 38, H. Kangassalo et al. (Eds.), *Information Modelling and Knowledge Bases VIII*

ISSN: 0922-6389

Intelligent Technologies – Theory and Applications

New Trends in Intelligent Technologies

Edited by

Peter Sinčák

*Center for Intelligent Technologies, Department of Cybernetics and AI,
Faculty of Electrical Engineering and Informatics,
Technical University of Kosice, Slovakia
<http://www.ai-cit.sk>*

Ján Vaščák

*Center for Intelligent Technologies, Department of Cybernetics and AI,
Faculty of Electrical Engineering and Informatics,
Technical University of Kosice, Slovakia
<http://www.ai-cit.sk>*

Vladimír Kvasnička

*Department of Mathematics, Slovak Technical University,
Bratislava, Slovakia
<http://math.chtf.stuba.sk/staff/kvasnicka/kvasnic.htm>*

Jiří Pospíchal

*Department of Mathematics, Slovak Technical University,
Bratislava, Slovakia
<http://math.chtf.stuba.sk/staff/pospichal/pospich.htm>*



Amsterdam • Berlin • Oxford • Tokyo • Washington, DC

© 2002, The authors mentioned in the Table of Contents

All rights reserved. No part of this book may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, without the prior written permission from the publisher.

ISBN 1 58603 256 9 (IOS Press)

ISBN 4 274 90512 8 C3055 (Ohmsha)

Library of Congress Control Number: 2002105620

Publisher

IOS Press

Nieuwe Hemweg 6B

1013 BG Amsterdam

The Netherlands

fax: +31 20 620 3419

e-mail: order@iospress.nl

Distributor in the UK and Ireland

IOS Press/Lavis Marketing

73 Lime Walk

Headington

Oxford OX3 7AD

England

fax: +44 1865 75 0079

Distributor in the USA and Canada

IOS Press, Inc.

5795-G Burke Centre Parkway

Burke, VA 22015

USA

fax: +1 703 323 3668

e-mail: iosbooks@iospress.com

Distributor in Germany, Austria and Switzerland

IOS Press/LSL.de

Gerichtsweg 28

D-04103 Leipzig

Germany

fax: +49 341 995 4255

Distributor in Japan

Ohmsha, Ltd.

3-1 Kanda Nishiki-cho

Chiyoda-ku, Tokyo 101-8460

Japan

fax: +81 3 3233 2426

LEGAL NOTICE

The publisher is not responsible for the use which might be made of the following information.

PRINTED IN THE NETHERLANDS

Foreword

For decades, everyone has been told that the field of Artificial Intelligence was intentionally and formally created at the famous Dartmouth meeting of 1956. Yet, almost fifty years later, Artificial Intelligence is nowhere to be found. This book is intended to help end this long, disappointing epoch of failure.

The failure of AI has its origins in an incorrect view of knowledge. Traditionally, knowledge has been viewed as a body of facts. Despite the existence of some obvious exceptions to it, such as neural network systems that were trained to drive a car by mimicking a human driver's performance (this is knowledge without the possession of any facts), this incorrect view has been overwhelmingly dominant. In reality, knowledge is an ability to react usefully to stimuli. Facts are rationalizations of these reactive behaviors and can only be obtained for a tiny subset of the instances of knowledge. Even when facts can be obtained for a body of knowledge, they rarely encompass the full extent of that knowledge (e.g., an extensive set of facts about mechanics still does not allow us to build a robot that can run up stairs). As is becoming ever more clear, knowledge is not facts. And AI will not arise from work in the traditional directions.

Another major blockage in the quest for AI is a radically flawed understanding of the nature of human and animal intelligence. This seems impossible, since the serious study of human nature has been underway at least since the time of Aristotle 2,350 years ago. Surely we have the basics correct. I think not.

For example, despite millennia of study, it is still commonly believed that our minds are like file cabinets: they store away knowledge and experiences for later use as we proceed through life, but themselves remain rather static and unchanged throughout adulthood (barring damage). The reality is probably much different. For example, when we move or change jobs, a large portion of our brain probably radically reorganizes.

Another example of our massive self misunderstanding is provided by considering one of the most profound propensities of all higher mammals: to segment the world into holistic objects. Our family cat Zeus can, under amazingly diverse situations, easily detect and visually segment almost any mouse-sized visual object and attack it; even though he has no idea what the object is. Similarly for auditory stimuli. This is one of the major defining characteristics of mammalian intelligence; yet it is so familiar that we are blissfully unaware of its profundity.

Next, consider human conversation. Surely, from an ethological standpoint, this constitutes one of those supremely impressive capabilities that make our species cognitively superior to all others. Yet intense conversation involves essentially no cognition whatsoever. Response follows challenge so quickly that it is all the neurons can do to merely propagate the auditory input through the speech circuits to the larynx. Each stage of processing has only one or two synapse times to act. There is no possibility of any significant cognition. Thus, again, our vision of ourselves is vastly flawed. Conversation is not an exalted confirmation

of our unique mental superiority. It is merely another example of a *reflex*. A glorified knee-jerk.

A truer picture of conversation is of a bipartite system. One part is reflexive and this handles the job of reflexively creating logical, context-, goal- and drive-sensitive responses to conversational input. The other part creates, maintains and improves this first part (and configures and prepares it before, and occasionally during, each conversation). This latter part is indeed capable of impressive cognition; but it takes place over tens of thousands of hours spread over tens of years. While linguists might say that all of this is well known; in fact it is not. Almost no one thinks of themselves this way.

Another gaping void in our self-understanding is sensory appreciation of the world. Not only are we forever exposed to the delusion of a high-resolution visual world around us, but we actually believe that all music and other sounds that our ears pick up are processed continuously by our auditory system. In fact, both our sensory and auditory systems are highly episodic and attentively focused (at any moment they almost always ignore all but one object within the visual or auditory scene). The image of an animal with senses that dart around, attentively dwelling on one item and then another is certainly not the way we view ourselves. Yet this is the animal we actually are.

The quest for AI must now turn to those elements that actually make us capable. This book is in the vanguard of that new direction. Adaptation to change, learning by doing, learning by education, self-organizing to an information environment, goal seeking, allowance for imprecision, and so on, are key elements of AI that this book reports progress on.

As we now proceed along the path to true AI, we may find the exalted pedestal upon which we have traditionally placed ourselves significantly de-elevated. The real difference between us and other mammals will turn out to be much smaller than we currently think. Most likely, however, we will also come to understand that it is our ongoing accumulation of codified knowledge and culture (including this book) that is the most important and noble aspect of our species.

Robert Hecht-Nielsen
San Diego, California
21 February 2002

Preface

What are Intelligent Technologies and what is the state of the art and the future of these technologies in Information Society? In general we can say that intelligent technology is technology working with machine intelligence (MI) tools. Machine Intelligence as an increasingly important part of Artificial Intelligence is a very progressive research domain supported by fast growing progress in computer technology. Expert systems, neural networks, fuzzy systems, evolutionary algorithms and other related technologies create the basic framework of machine intelligence tools. The number of current applications in industry and application potential of machine intelligence and intelligent technology is growing by speed-up of computer technology progress.

The main challenge of intelligent technologies is to provide technology with some of the following features (selected features) :

- Ability to learn from example – learning technology (it is a higher degree of adaptive technology)
- Ability to extract knowledge from human – by interaction with human to extract his knowledge and implement the knowledge-base for future use
- Ability to make function interpolation and extrapolation – more specifically an ability to approximate unknown functions which are not described analytically but by data
- Ability to reveal some correlations in data which cannot be seen by other means
- Ability to interface with verbal expression and process it for future purpose
- Ability to provide an effective search in domain space to find optimal parameters
- Ability to store the knowledge base in the form of incremental adjustments with non-forget approach of the process

The future of Information Society is clearly in embedding Machine Intelligence into the information technology. There is an extensive support from the EU within the IST program to support these technologies. The MI tools should play an important role in these expectations.

This book is presenting the papers from Euro-International Symposium on Computational Intelligence which was held in Kosice, Slovakia. Approximately 100 people gathered in this historical town with the aim of discussing the important aspects of Computational Intelligence in current and future technologies. The progress is mainly noticeable in applications of MI tools based on evolutionary computing because of significant change and speed/up of computer technology. Basically the progress of hybrid technologies is still evident as the only effective approach to problem oriented technologies. Customers do not care anymore about the question which technologies were used (neural networks, fuzzy systems, evolutionary systems etc.) but they clearly need better and advanced technology. Since the Machine Intelligence communities have lots of valuable arguments based on a practical experience for the combined use of the methods discussed, it is a general belief that MI will play an increasingly important role in current and future intelligent technologies.

We thank Prof. Hecht-Nielsen for a foreword to these proceedings and his comments and opinions about Intelligent Technologies. It is very interesting reading since he is a witness of MI tools success story in technologies applied at HNC, one of the world leading commercial companies promoting these technologies.

Also we would like to thank the publishing team of IOS Press for publishing this book and spreading its ideas to the MI community and wider public throughout the world.

Next, we thank students from the Center for intelligent technologies at Technical University of Kosice, Slovakia for their tireless effort in preparing and formatting this manuscript for publication. Finally we thank all the sponsoring institutions for their support: in addition to the EU, we also received funding from the Ministry of Education of the Slovak Republic, International Neural Network Society, European Neural Network Society and other organizations.

March, 2002

Peter Sinčák
Ján Vaščák

Vladimír Kvasnička
Jiří Pospíchal

Center for Intelligent Technologies
Technical University of Kosice
Kosice, Slovakia

Department of Mathematics
Slovak Technical University
Bratislava, Slovakia

Contents

Foreword, <i>R. Hecht–Nielsen</i>	v
Preface	vii

I. Neural Networks

Superconvergence Concept in Machine Learning, <i>P. Géczy, S. Amari and S. Usui</i>	3
Application of the Parallel Population Learning Algorithm to Training Feed-forward ANN, <i>I. Czarnowski and P. Jędrzejowicz</i>	10
Markovian Architectural Bias of Recurrent Neural Networks, <i>P. Tiño, M. Čerňanský and L. Beňušková</i>	17
SOFM Training Speedup, <i>M. Jiřina</i>	24
Generalized Forecasting Sigma–Pi Neural Network, <i>Y. Bodyanskiy, V. Kolodyazhniy and N. Kulishova</i>	29
Human Centered Intelligent Robots Using “Ontological Neural Network”, <i>T. Yamaguchi, E. Sato and S. Watanabe</i>	34
Trajectory Bounds of Solutions of Delayed Cellular Neural Networks Differential Systems, <i>J. Buša and V. Pírč</i>	40
From Plain to Modular Topology: Automatic Modularization of Structured ANNs, <i>R. Jakša</i>	48
Neural Structure as a Modular Developmental System, <i>E. Volná</i>	55
Prediction Systems Based on FIR Neural Network with Correction of Error, <i>D. Novotný and P. Sinčák</i>	61
Complex Cognitive Phenomena – Challenge for Artificial Intelligence, <i>M. Dobeš</i>	68
Convergence of Action Dependent Dual Heuristic Dynamic Programming Algorithms in LQ Control Tasks, <i>D. Krokavec</i>	72

II. Fuzzy Systems

A Prototype-centered Approach to Adding Deduction Capability to Search Engines – The Concept of Protoform, <i>L.A. Zadeh</i>	83
Compromise Approach to Neuro–Fuzzy Systems, <i>L. Rutkowski and K. Cpałka</i>	85
On Uninorms and their Residual Implicators, <i>J. Fodor and B. De Baets</i>	91
Adaptation of Fuzzy Controllers Based on Incremental Models, <i>J. Vaščák and K. Hirota</i>	98
Optimal Order of Convergence for α -cut Based Fuzzy Interpolators, <i>D. Tikk</i>	105
Properties of Fuzzy Controller with Conditionally Firing Rules, <i>M. Navara and J. Št'astný</i>	111
An Algorithm for the Network Steiner Tree Problem with Fuzzy Edge Lengths, <i>M. Šeda</i>	117
A Method For Learning of Hierarchical Fuzzy Systems, <i>R. Nowicki, R. Scherer and L. Rutkowski</i>	124
Risk Analysis in Fuzzy Flow Networks, <i>R.V. Tyshchuk</i>	130
Fuzzy Model of Student's Behaviour Generated by Means of GMDH Algorithms, <i>P. Náplava and M. Šnorek</i>	135

III. Evolutionary Computations and Miscellaneous Biologically Inspired Systems

Evolving Connectionist Systems for Modelling and Implementation of Adaptive, Intelligent Interactive Systems, <i>N. Kasabov</i>	141
The Population Sizing Problem in (P)GAs: Experiments, <i>Z. Konfršt and J. Lažanský</i>	146
A Constraint Handling Method for the Differential Evolution Algorithm, <i>J. Lampinen</i>	152
Competing Heuristics in Evolutionary Algorithms, <i>J. Tvrđík, L. Mišík and I. Křivý</i>	159
Evolvable Computational Machines: Formal Approach, <i>L. Sekanina</i>	166
Steady-state Evolutionary Algorithm with an Operator Family, <i>L. Gacôgne</i>	173
Genetic Algorithms with Changing Criterion Functions, <i>I. Sekaj</i>	183
Visualization of Topic Distribution from Document Sequence on Web, <i>Y. Takama and K. Hirota</i>	189
A Tool for Data Mining Support, <i>M. Hubal' and P. Bednár</i>	196
Effects of Cuing on Perceived Location of Auditory Sources, <i>N. Kopčo and B. Shinn-Cunningham</i>	201
Stability of Eco-problem Solving in Positional Problems, <i>I. Kapustík and J. Hercek</i>	207
Particle Swarm Optimization Method for Constrained Optimization Problems, <i>K.E. Parsopoulos and M.N. Vrahatis</i>	214
Various Approaches in Modeling of Algae Population, <i>J. Chval and M. Palko</i>	221
Estimation Distribution Algorithm for Mixed Continuous–Discrete Optimization Problems, <i>J. Očenášek and J. Schwarz</i>	227
On Dimensionality Reduction of High Dimensional Data Sets, <i>B. Chizi, A. Shmilovici and O. Maimon</i>	233
Neocognitron and Its Applications in Image Processing, <i>M. Šamulka, P. Kostelník and M. Hudec</i>	239
The Experimental Study of the Competing Co-evolution Using Predator–Prey Tasks, <i>G. Gebeová, M. Hudec, P. Kostelník and V. Kováč</i>	245
Two-dimensional Hologram Model and Computer Simulation of Patterns Processing, Storing and Restoring, <i>A. Profir, E. Boian and C. Zelinschi</i>	251

IV. Applications

Evolutionary Human Support Agents and Applications, <i>T. Yamaguchi, A. Sannoh and M. Iori</i>	255
Distributed Learning in Behavioral Based Mobile Robot Control, <i>P. Kostelník, M. Hudec and M. Šamulka</i>	262
Mass Approximation from Noise Signals of Loose Part Monitoring Systems, <i>Š. Fígedy and M. Hrehuš</i>	272
Visualising Genetic Algorithms: A Way through the Labyrinth of Search Space, <i>M. Mach and Z. Zetková</i>	279
Genetic Algorithm Optimization of the Dataflow Processor, <i>M. Turčaník and M. Liška</i>	286
Pareto-optimization with EA in Application of Centrifuges Design, <i>X. Feng and J. Strackeljanc</i>	291
Neuro-tomography: A New Approach to Image Reconstruction from Projection, <i>R. Cierniak</i>	298

New Approach to Estimation of Pulse Coupled Neural Network Parameters in Image Recognition Process, <i>R. Forgáč and I. Mokriš</i>	304
Prediction of Gross Domestic Product Development on the Basis of Frontal Neural Networks, Genetic and Eugenic Algorithms, <i>V. Olej</i>	309
A Fuzzy Logic Approach for the Condition Monitoring of Rotating Machines, <i>J. Strackeljan</i>	315
Neural Networks for SISD Real Time Control, <i>M. Čapkovič and Š. Kozák</i>	321
Composite Fuzzy Measure and its Application to Investment Decision Making, <i>T. Kaino and K. Hirota</i>	329
Prediction of the Index Fund on the Basis of Fuzzy Inference Systems, <i>V. Olej and M. Fedurco</i>	336
Evolutionary Algorithms Designer – Graphical Way of Implementing Evolutionary Algorithms, <i>P. Macej</i>	342
Author Index	345

This page intentionally left blank

I. Neural Networks

This page intentionally left blank

Superconvergence Concept in Machine Learning

Peter GÉCZY, Shun-ichi AMARI, Shiro USUI

RIKEN Brain Science Institute,

2-1 Hirosawa, Wako-shi, Saitama 351-0198, Japan

Toyohashi University of Technology,

1-1 Hibarigaoka, Tempaku-cho, Toyohashi 441–8580, Japan

Abstract. We analyze a class of learning algorithms with the fastest convergence rates - superconvergence rates. Superconvergence concept is independent of the order of an algorithm, but depends on the used metric. Analysis of behavioral dynamics of superconvergent algorithms in the neighborhood of critical points with nondegenerative singularities on general metric spaces is provided. Presented theoretical material is illustratively demonstrated and applied to learning multilayer perceptron (MLP) networks.

1 Motivation

It has been observed that the manifold of stochastic MLP networks with the coordinate system given by the network parameters has a parameter space with Riemannian structure [1]. It implies that the conventional gradient in the line search gradient based learning algorithms represents the steepest direction of the loss function only when the coordinate system is orthonormal. This has led to the development of the Natural Gradient (NG) learning technique utilizing Riemannian metrics on the parameter space. It has been observed that the NG learning algorithm features Fisher efficiency and faster convergence [1]. Impediment of NG learning is high computational complexity resulting from the calculation of the Fisher information matrix and its inverse, or approximations [2].

If the search direction can be determined (e.g. using NG), the issue of the length of the progress still remains. We focus on the general analysis of the norm of iterative update in the context of superconvergence rates on an arbitrary metric space.

2 Formulation of Approach

Connectionistic models adjust a number of real-valued parameters when tuning the system to achieve satisfactory performance. Parameter space \mathcal{S} represents a subset of real space \mathbb{R}^{N_F} , where N_F is a number of free parameters. Let (\mathcal{S}, ρ) be a metric space with a metric ρ defined on a set $\mathcal{S} \subset \mathbb{R}^{N_F}$. Each element of $u \in \mathcal{S}$ corresponds to the particular setting (or state) of a connectionistic system.

Performance of a system is evaluated with respect to an objective function E that is closed and bounded on \mathcal{S} ; and is sufficiently smooth: $E \in C^\infty$. Function $E : \mathcal{S} \rightarrow \mathbb{R}$ incorporates

a mapping of a system and for each parameter vector $u \in \mathcal{S}$ produces a real value that describes system's performance. In practice are often used error functions, such as the mean square error (MSE), or information theoretic measures.

Learning refers to an iterative process of finding an appropriate parameter vector $u^* \in \mathcal{S}$ that optimizes (either minimizes or maximizes) the objective function E given a finite number of data elements drawn from a data pool with an arbitrary distribution. Iterative process, called learning (or training) algorithm, generates a sequence of points $\{u^{(k)}\}$ in the parameter space \mathcal{S} that converges to a point u^* in a solution set $\Xi \subset \mathcal{S}$ for which the objective function E satisfies certain condition, e.g. $E(u^*) \leq c$, $c \in \mathbb{R}$; in the further text $E(u^*)$ will be denoted as E^* .

Consider a convergent sequence of points $\{u^{(k)}\} \rightarrow u^* \in \Xi$ generated by an algorithm \mathcal{A} such that each iteration is expressed as,

$$u^{(k+1)} = u^{(k)} + \Delta u^{(k)} \text{ or } u^{(k+1)} = u^{(k)} - \Delta u^{(k)}, \quad (1)$$

where $\Delta u^{(k)}$ is a function of $E(u^{(k)})$ (further denoted as $E^{(k)}$).

Definition 1. It is said that algorithm \mathcal{A} has convergence rates a of the i -th order on a metric space (\mathcal{S}, ρ) if it generates a convergent sequence of points $\{u^{(k)}\} \rightarrow u^*$ such that the following holds.

$$a = \limsup \frac{\rho(u^* - u^{(k+1)})}{[\rho(u^* - u^{(k)})]^i}; \quad i \in \mathbb{N}^+ \quad (2)$$

If $a = 0$, it is said that an algorithm \mathcal{A} has super- i -th convergence rates. If irrespective of the order, the term superconvergence rates is used. ■

Note that the fastest convergence rates for given order i are the super- i -th convergence rates.

3 Superconvergence Rates

Theorem 1. Let $E \in C^\infty$ be a function closed and bounded on a metric space (\mathcal{S}, ρ) and \mathcal{A} be an algorithm generating a convergent sequence of points $\{u^{(k)}\}$ described by (1) with superconvergence rates. The following holds.

$$\rho(\Delta u^{(k)}) = \left| \frac{E^* - E^{(k)}}{\rho(\nabla E^{(k)})} + \varepsilon_{\Delta_k} \right|, \quad (3)$$

where

$$\varepsilon_{\Delta_k} = \frac{R_{n \geq 2}}{\rho(\nabla E^{(k)})}, \quad (4)$$

and $R_{n \geq 2}$ denotes the second and higher order residual terms of generalized Taylor expansion of E around u^* . ■

The norm of the update term $\Delta u^{(k)}$ is expressible by relatively simple formula (3). In practice very often the second and higher order terms are neglected when designing learning

algorithms. Then expression (4) corresponds to the caused error to the norm of the iterative update.

Information theoretic approach introduced arguments [1] that the space of stochastic neural networks is not Euclidean. The question arises: What metrics are generally suitable for superconvergent learning algorithms? Implying from Theorem 1, for a given model and objective function E , the most suitable metric ρ is the one that minimizes ε_{Δ_k} . Selection of metric ρ can be viewed as a minimization task:

$$\min_{\rho} \left[\frac{R_{n \geq 2}}{\rho(\nabla E^{(k)})} \right]. \quad (5)$$

Assume that the metric ρ was chosen according to (5) and the term ε_{Δ_k} becomes negligible. From (3) follows:

$$\rho(\Delta u^{(k)}) \approx \left| \frac{E^* - E^{(k)}}{\rho(\nabla E^{(k)})} \right|. \quad (6)$$

Considering the update formula for line search algorithms (whether first, second, or generally i -th order),

$$\Delta u^{(k)} = \alpha^{(k)} s^{(k)}; \alpha^{(k)} \in \mathfrak{R}, \quad (7)$$

where $\alpha^{(k)}$ is the step length and $s^{(k)}$ is the search direction, it immediately follows:

$$|\alpha^{(k)}| \approx \left| \frac{E^* - E^{(k)}}{\rho(\nabla E^{(k)})\rho(s^{(k)})} \right|. \quad (8)$$

Update formula of conjugate search algorithms,

$$\Delta u^{(k)} = \alpha^{(k)} s^{(k)} + \beta^{(k)} s^{(k-1)}; \alpha^{(k)}, \beta^{(k)} \in \mathfrak{R}, \quad (9)$$

requires more complex convergence analysis that finally leads to the following:

$$|\alpha^{(k)}| \approx \left| \frac{E^{(k)}}{\rho(\nabla E^{(k)})\rho(s^{(k)})} \right|; |\beta^{(k)}| \approx \left| \frac{E^*}{\rho(\nabla E^{(k)})\rho(s^{(k-1)})} \right|. \quad (10)$$

Importance of expressions (8) and (10) is threefold: **I.** introduction of general formulas for all line search algorithms with respect to metric ρ ; **II.** reduction of the line search subproblem to a single-step calculation; **III.** automatic determination of the appropriate step length and/or momentum term at each iteration of an algorithm.

4 Specific Critical Points and Singularities

Further analysis reveals additional interesting features related to the dynamics of expressions (8) and (10) in the close neighborhood of critical points with nondegenerative singularities.

Saddle Points and Extremes These are the simplest critical points at which the gradient $\nabla E^{(k)}$ is equal to 0-vector. If for the metric ρ holds: $\rho(\vec{0}) = 0$, then $\alpha^{(k)}$ in (8), (10) and $\beta^{(k)}$ in (10) become undefined. This holds e.g. for l-metrics. Though this situation in practice very rarely occurs, it requires attention. Straight solution to this problem is to set boundaries $\alpha^{(k)} \in \langle \alpha_{\min}, \alpha_{\max} \rangle$ and $\beta^{(k)} \in \langle \beta_{\min}, \beta_{\max} \rangle$, and to use the maximum values of α and

β , if the algorithm did not reach the solution set Ξ . Interesting is to observe behavior of $\alpha^{(k)}$ and $\beta^{(k)}$ in the neighborhood of these points. If $\rho(\nabla E^{(k)})$ (or $\rho(s^{(k)})$, $\rho(s^{(k-1)})$) approaches 0, the denominator of expressions for $\alpha^{(k)}$ and $\beta^{(k)}$ approaches 0. Assume the algorithm \mathcal{A} is not reaching the optimum point, that is $E^* - E^{(k)} \neq 0$ and/or $E^* \neq 0$, then the terms $\alpha^{(k)}$ and $\beta^{(k)}$ take large values which results in the dramatic parameter update. This mechanism enables algorithms utilizing $\alpha^{(k)}$ (8), (10) and $\beta^{(k)}$ (10) to automatically escape from local minima. Illustrative demonstration of this important feature is presented in the following paragraph.

We constructed simple first order steepest descent algorithm \mathcal{A} utilizing update ex-

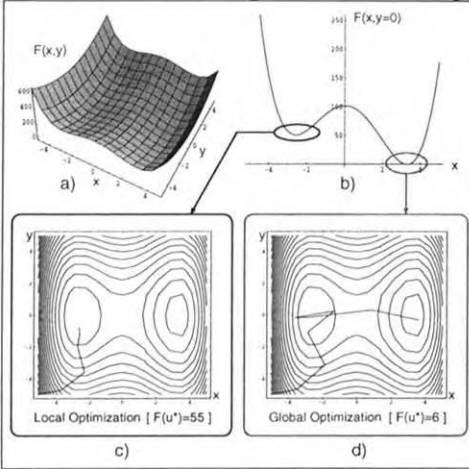


Figure 1: Illustration of escape from the local minimum.

pressions (7) and (8). Used metric ρ was l_2 norm. Algorithm \mathcal{A} was applied to minimizing function $F(x, y) = x^4 - x^3 + 17x^2 + 9y^2 + y + 102$. The function is quadratic with respect to 'y' and of the 4th order with respect to 'x'. 3D plot of the function is shown in chart a) of Figure 1. Function $F(x, y)$ has two minima; one global and one local. Chart b) of Figure 1 shows the position of local and global minima at the cutting plane $y = 0$. The starting point of the optimization was $[x, y] = [-5, -5]$. In the first case, the expected stopping value was set $F(u^*) = 55$. Progress of algorithm \mathcal{A} is shown in contour plot c) of Figure 1. The algorithm stopped after 5 iterations reaching the value $F(u^*) \leq 55$ in the local minimum. In the second case, the expected value was set to 6. The optimization progress of the algorithm is shown in contour plot d) of Figure 1. Starting from the point $[-5, -5]$, the algorithm initially converged to the local minimum. Small values of the gradient norm in the area around the local minimum, and discrepancy $|F(u^{(k)}) - F(u^*)|$, led to large values of $\alpha^{(k)}$ and dramatic parameter update. This caused the algorithm to jump out of the local minimum and finally (after escaping from the local minimum) in a single step to reach the appropriate value of F in the neighborhood of the global minimum.

E^* Contours Points at which function E takes values E^* represent another important case for analysis. Here the numerator of $\alpha^{(k)}$ (8) is zero and if algorithm utilizes only the step length update according to (8), it stops. This is in accordance with the stopping condition given by the value of E^* . In the close neighborhood of E^* contours algorithm utilizing only $\alpha^{(k)}$ (8) update progresses slowly, since $\alpha^{(k)} \rightarrow 0$. To speed up the progress requires to introduce additional update mechanism. Sufficient and simple solution is to use in the numerator of $\alpha^{(k)}$ the expression $E^* - E^{(k)} - q$; $q \in \mathbb{R}$, where q is a small real number. Another solution is to use α_{min} if α is considered to be bounded.

Completely Flat Regions A set $\mathcal{T} \subset \mathcal{S}$ of points $u \in \mathcal{S}$ such that $\nabla E(u) = \bar{0}$ and for which there exists a neighborhood $O_u \subset \mathcal{S}$ such that for each point $v \in O_u$ the gradient $\nabla E(v)$ is zero vector is a completely flat region. If the metric ρ has the property $\rho(\bar{0}) = 0$, similar situation as that of the saddle points and extremes occurs. In this case it is not just a single point, but the whole (measurable) region. In the neighborhood of flat region, where

$\rho(\nabla E^{(k)}) \rightarrow 0$, algorithm utilizing step length updates (8) and (10) progresses rapidly. This accounts for the potential of an algorithm to escape flat regions and to progress fast on nearly-flat-regions, where $\rho(\nabla E^{(k)}) \approx 0$.

4.1 Example of Adaptable Step Length Dynamics

Typical dynamics of adaptable $\alpha^{(k)}$ are demonstrated on the task of optimizing artificial neural network parameters by using first order steepest descent algorithm \mathcal{A} with updates according to (7) and (8) and metric $\rho \equiv l_2$. A neural network had the configuration 4–3–1 with sigmoidal nonlinear elements and was trained on the Lenses data set [3]. Stopping condition was: $\text{MSE} < 5 \cdot 10^{-2}$.

After the initial progress (approximately 5 cycles) the network located the flat area of the

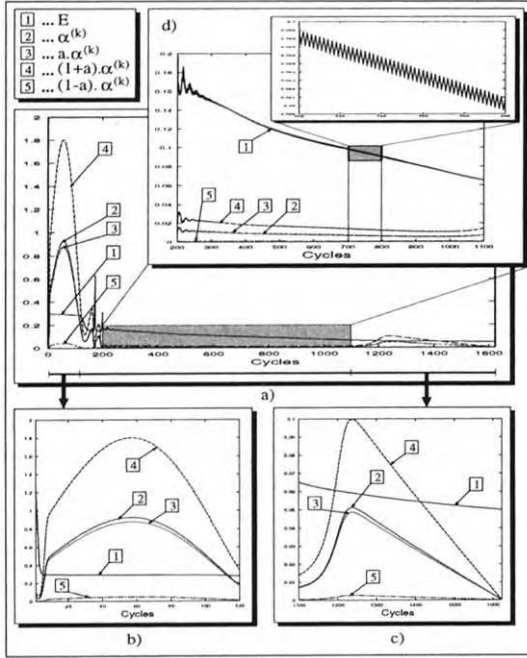


Figure 2: Typical dynamics of adaptable step length $\alpha^{(k)}$.

surface, on which the network was progressing, caused high values of derivatives with respect to the parameters in which the error surface sharply decreased and low values of derivatives with respect to the parameters where the error surface was relatively flat. The result was that the gradient pointed to the other side of the multidimensional ravine rather than to its minimum. In such situations it is essential for a steepest descent technique to lower the values of $\alpha^{(k)}$ so as to reach the bottom of the ravine in the fewest number of cycles. Again, the theoretically derived expressions for dynamically adaptable $\alpha^{(k)}$, depicted by curves [2], [3], [4], [5], clearly follow this necessity.

In the final phase of a network's training, starting from cycle 1100, the eccentricity of the attractor basin around the terminal attractor point u^* slightly relaxed. This led to a relative balance of first partial derivatives with respect to the free parameters. Gradient ∇E pointed almost directly to the optimum point u^* . To reach the terminal attractor faster, the optimization procedure automatically increased the $\alpha^{(k)}$. This behavior is depicted in chart c).

After approximately 150 cycles the network reached a strongly eccentric region of the error surface. Details of this phase (120 cycles) are displayed in chart b). The flatness of the surface results in very low values of the first derivatives and hence the values of the gradient, ∇E . Since the search direction $s^{(k)}$, in steepest descent, is determined by the gradient $\nabla E(u^{(k)})$, very slow progress occurs. The natural action in this situation is to speed up the progress by increasing $\alpha^{(k)}$. As it can be seen from curves [2], [3], [4], and [5], the dynamically adaptable automatically follows this rule.

After approximately 150 cycles the network reached a strongly eccentric region of the error surface. Details of this phase, starting from cycle 200, are shown in chart d). The slight oscillations are enlarged in the upper right chart of chart d). Eccentricity of the

5 Illustrative Simulations

We implemented the first order conjugate gradient algorithm \mathcal{A} according to (9) and (10). Chosen metric ρ was l_2 norm. The algorithm was compared to the relevant techniques within the same class, that is, the first order methods and to the pseudo-second order method called Kick-Out [4]. It has been observed that Kick-Out outperforms conventional learning algorithms and their variations.

Method The effectiveness of the algorithm is practically demonstrated on five tasks represented by the following data sets: Lenses [3], Glass, Monks 1 [5], Monks 2 [5], and Monks 3 [5]. The presented algorithm was applied to training various MLP networks. Neural networks' performance was optimized according to MSE. Stopping criterion was the value of the expected error.

Setup In the case of the Lenses data set, a neural network had configuration 4–3–1. Expected error was set to $5 \cdot 10^{-2}$. In the Glass problem, a network was configured as: 9–5–1 and the expected error was equal to 0.35. Finally, for Monks 1, 2, and 3 problems a neural network structure was set as: 6–3–1 and the expected error was equal to 0.103. All network configurations used sigmoidal hidden units. Network's weights were initialized randomly in the interval $< -0.1, 0.1 >$, corresponding to the steepest region of the sigmoidal transfer function of the hidden units. In case network's error did not converge to the value less than or equal to the expected error within 20000 cycles, the training process was terminated.

Experiment Outline First, the standard BP with the constant term α ranging from 0.1 to 0.9 in 0.1 increments and Kick-Out algorithm were trained to determine the best parameter setting. Remaining experiments were performed with the best observed parameter values for BP and Kick-Out (BP: in Monks 1 case $\alpha = 0.8$, and for all other data sets $\alpha = 0.9$). Kick-Out's additional parameters were set as follows: $\kappa = 0.0001$, $\phi = 0.9$ and $\tau = 0.7$. The momentum term ranging from 0.1 to 0.7 in 0.1 increments was used. Kick-Out and BP with the momentum term and the best value of (denoted in further text as BPM) were compared to the proposed algorithm \mathcal{A} . For a given setting of learning rate and momentum term the simulations were run 10 times for different randomly initialized weights.

Table 1: Comparison between the derived algorithm \mathcal{A} , Kick-Out and standard BPM

Momentum Term β		0.1	0.2	0.3	0.4	0.5	0.6	0.7	Task AVERAGE
Lenses	BPM	2.05	1.26	1.11	1.88	14.66	20.62	34.84	10.92
	KO	1.5	1.11	1.03	1.25	5.31	5.72	6.11	3.15
Glass	BPM	2.24	2.14	1.96	1.85	1.70	1.61	1.43	1.85
	KO	1.31	1.3	1.24	1.36	1.28	1.13	1.01	1.23
Monks 1	BPM	2.46	2.14	1.76	1.71	2.16	8.19	14.81	4.75
	KO	1.32	1.26	1.21	1.17	1.33	2.21	3.15	1.66
Monks 2	BPM	1.01	0.96	1.17	2.55	8.31	37.17	48.31	14.21
	KO	0.97	0.96	1.11	1.35	3.24	5.89	6.73	2.89
Monks 3	BPM	1.73	1.68	1.56	2.50	6.01	9.93	12.87	5.18
	KO	1.25	1.21	1.18	1.77	2.31	2.66	3.24	1.95

Results The values in Table 1 represent ten-run-averages. Convergence speed increase values in the table indicate how many times the proposed algorithm \mathcal{A} converged faster than BPM and Kick-Out techniques in terms of the number of cycles required to decrease the mean square error E of a neural network below the value of the expected error. It is clear that the proposed algorithm \mathcal{A} converged substantially faster. The algorithm was on the average form

approximately 2 to 14 times faster than BPM and 2 to 11 times faster than Kick-Out. The proposed algorithm converged each time, whereas BPM for some initial setting of weights and parameters could not converge even after 20000 cycles.

6 Conclusions and Future Concept

The derived theoretical construct of superconvergence was practically applied to the first order methods for the purpose of 'classical' illustration. It should be noted that there is a direct applicability to the second order methods (and generally to the i -th order methods). The only difference is the determination of the search direction and consideration of the second order information. It can also be proved that the presented first order methods are convergent - with superlinear convergence rates, and have linear computational complexity $O(N_F)$ [6], [7]. Analogous proofs can be provided for the second order methods. Dynamic behavior in the neighborhood of critical points can be utilized for structural adaptation of neural networks during the learning without harming the convergence speed. This task is under investigation.

Acknowledgment

The authors would like to thank Dr. Naohiro Toda of Aichi Prefectural University and Dr. Taichi Hayasaka of BPEL at Toyohashi University of Technology for their useful comments.

References

- [1] S. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10:251–276, 1998.
- [2] S. Amari, H. Park, and K. Fukumizu. Adaptive method of realizing natural gradient learning for multilayer perceptrons. *Neural Computation*, 12:1399–1409, 2000.
- [3] J. Cendrowska. Prism: An algorithm for inducing modular rules. *International Journal of Man-Machine Studies*, 27:349–370, 1987.
- [4] K. Ochiai, N. Toda, and S. Usui. Kick-Out learning algorithm to reduce the oscillation of weights. *Neural Networks*, 7(5):797–807, 1994.
- [5] J. Wnek and R. S. Michalski. Comparing symbolic and subsymbolic learning: Three studies. In R. S. Michalski and G. Tecuci, editors, *Machine Learning: A Multistrategy Approach*, volume 4, San Mateo, 1993. Morgan Kaufmann.
- [6] P. Géczy and S. Usui. Novel first order optimization classification framework. *IEICE Transactions on Fundamentals*, E83-A(11):2312–2319, 2000.
- [7] P. Géczy and S. Usui. Superlinear conjugate gradient method with adaptable step length and constant momentum term. *IEICE Transactions on Fundamentals*, E83-A(11):2320–2328, 2000.

Application of the Parallel Population Learning Algorithm to Training Feed-forward ANN

Ireneusz Czarnowski
Piotr Jędrzejowicz

*Department of Information Systems, Gdynia Maritime University,
Morska 83, 81-225 Gdynia, Poland irek, pj@wsm.gdynia.pl*

Abstract. The paper investigates application of the new metaheuristic called population learning algorithm (PLA) to training feed-forward artificial neural networks. The paper introduces the population learning algorithm and proposes its parallel implementation developed with a view to train ANN solving the two-spiral problem. Computational experiment results show high effectiveness and good quality of the suggested approach.

Keywords: *Population learning algorithm, Neural networks, Training algorithm, Parallel computations*

1 Introduction

Artificial neural networks are, nowadays, being used for solving a wide variety of real-life problems like, for example, pattern recognition, prediction, control, combinatorial optimization or classification. Main advantages of the approach include ability to tolerate imprecision and uncertainty and still achieving tractability, robustness, and low cost in practical applications. Since training a neural network for practical application is often very time consuming, an extensive research work is being carried out in order to accelerate this process. Another problem with ANN training methods is danger of being caught in a local optimum. Hence, researchers look not only for algorithms that train neural networks quickly but rather for quick algorithms that are not likely, or less likely, to get trapped in a local optimum [1] [2] [10].

One of the attractive and widely investigated approaches is to train neural networks using the genetic algorithm as a learning tool (see for example [2] [5] [7]). In this approach a coded solution of the ANN training problem known as an individual represents a set of values characterizing neural network configuration. Population of individuals evolves through application of genetic operators - crossover and mutation. After a number of generations have been produced, population of individuals is expected to contain good solutions to the ANN training problem. The approach is known to produce good results in terms of the respective ANN quality. Unfortunately the discussed training process is often ineffective from the point of view of computational time requirements.

In this paper we investigate application of the recently proposed population-based meta-heuristic known as the population-learning algorithm (PLA) [4]. To test and evaluate application of the PLA to training neural networks the two-spiral problem has been chosen. Training ANN solving the two-spiral problem is considered as one of the most difficult benchmark problems in the field of training feed forward networks. The problem description together with the respective data sets can be obtained from <http://www.cae.wisc.edu/~ece602/data/CMU-benchmark>.

The paper briefly presents an idea of the population learning algorithm and gives a detailed description of the PLA implementation designed for training ANN solving the two-spiral problem. The proposed implementation involves a real representation of individuals and a parallel computational environment.

2 PLA and Feed-forward ANN Training

2.1 Concept of the PLA

Continuing research interest is focused on algorithms that rely on analogies to natural processes and phenomena. An important subclass of the NPB (natural processes based) class form population based methods, which also are called evolutionary algorithms.

Population based methods handle a population of individuals that evolves with the help of information exchange and self-improvement procedures. It should be noted that many different algorithms could be described within this framework. The best-known population-based algorithms are genetic algorithms, evolution strategies, evolution programming, scatter search, adaptive memory algorithms, and ant systems.

Population learning algorithms (PLA) also handle a population of individuals. An individual could be a solution of the considered problem, a part of the solution or, in fact, any other object that can be somehow transformed, perhaps in conjunction with other objects, into a solution.

Initially, a massive population of individuals, called the initial population, is generated. The number of individuals in the initial population should be sufficient to represent adequately the whole space of feasible solutions. Sufficient number of individuals relates to the need of covering the neighbourhood of all of the local optima. Adequate representation of these neighbourhoods is related to the need of assuring that the improvement process, originated at the initial stage should be effective enough to carry at least some individuals to the highest stages of learning. The number of individuals for the particular PLA implementation is, usually, set at the fine-tuning phase.

Generating the initial population could be, simply, based on some random mechanism assuring the required representation of the whole feasible solution space. Once the initial population has been generated, individuals enter the first learning stage. It involves applying some, possibly basic and elementary, improvement schemes or conducting simple learning sessions. The improved individuals are then evaluated and better ones pass to the subsequent stage. A strategy of selecting better or more promising individuals must be defined and duly applied. At the following stages the whole cycle is repeated. Individuals are subject to improvement and learning, either individually or through information exchange, and the selected ones are again promoted to a higher stage with the remaining ones dropped-out from the process. At the final stage the remaining individuals are reviewed with a view to selecting a solution to the problem at hand.

Learning process at early stages can be run in parallel. Individuals are then grouped into classes with possibly different curricula, which are different improvement schemes. At certain level the best from all groups join together to form higher-level groups where improvement and learning process are still carried in parallel. At some stage selected individuals are brought together to complete education.

At different stages of the process, different improvement schemes and learning procedures are applied. These as a rule, gradually become more and more sophisticated and time consuming as there are less and less individuals to be taught.

2.2 ANN Training Problem

ANN training involves finding a set of weights of links between neurons such that the network generates desired output signals. Training process is considered as adjusting values of these weights using a set of training patterns showing the desired ANN behaviour. In other words, given a set of training patterns consisting of input-output pairs $\{(u_1, d_1), (u_2, d_2) \dots (u_n, d_n)\}$ and an error function $e = (W, U, D)$, training process aims at minimizing learning error $E(W)$:

$$E(W) = \min_W \sum_{i=1}^N e(W, u_i, d_i) \quad (1)$$

where W is a set of the respective weights, U - is a set of inputs and D - a set of outputs.

One of the commonly used measures (error functions) is the squared-error function in which $e(W, u_i, d_i) = (d_i - f(u_i, W))^2$, where $f(u_i, W)$ represents ANN response for input u_i , and weights W . The measure is also used in this paper. The quality of trained network is assessed by its error on a given set of training patterns and on a given set of test patterns. [10]

2.3 The Two-spiral Problem

The two-spiral problem has been chosen as a benchmark used in this paper to evaluate application of the PLA to ANN training. ANN solving the two-spiral problem has two real-valued inputs corresponding to the x and y coordinates of the point, and one binary target output that classifies the point as belonging to either of the two spirals coiling three times around the origin.

The two spirals are constructed from 97 points given by the respective co-ordinates from the training set. Solving the two-spiral problem appears to be a very difficult task for back-propagation networks [4] [9].

2.4 PLA Implementation

In this paper we propose using population-learning algorithm to train artificial neural networks. PLA implementation for training ANN solving the two-spiral problem (PLA-2S) is based on the following assumptions:

- Parallel PLA scheme with convergence is used.
- An individual is a vector of real numbers from the interval $[-10, 10]$, each representing a value of weight of the respective link between neurons in the considered ANN.

- The initial population of individuals is generated randomly.
- There are three learning/improvement procedures - non-uniform mutation, single point crossover, and application of the gradient adjustment operator.
- There is a common selection criterion for all stages. At each stage, individuals with fitness below the current average are rejected.

The proposed parallel PLA scheme is based on the co-operation between the master worker whose task is to manage computations and a number of slave workers, who act in parallel performing computations requested by the master. The approach allows a lot of freedom in designing population-learning process. The master worker is managing communication flow during the population learning. It allocates computational tasks in terms of the required population size and number of iterations as well as oversees information exchange between slaves. The latter involves upgrading, at various learning stages, current populations maintained by all slaves with globally best individuals.

The following features characterize the proposed parallel PLA-2S:

- Master worker defines number of slave workers and size of the initial population for each of them.
- Each slave worker uses identical learning/improvement procedures.
- Master worker activates parallel processing.
- After completing each stage workers inform master about the best solution found so far.
- Master worker compares the received values and sends out the best solution to all the workers.
- Master worker can stop computations if the desired quality level of the objective function has been achieved. This level is defined at the beginning of computations through setting the desired value of the mean squared error on a given set of training patterns.
- Slave workers can also stop computations if the above condition has been met.
- The computations are carried out for the predefined number of iterations (number of stages).

3 Computational Experiment Results

To evaluate the parallel PLA-2S computational experiment has been carried out. It involved training a dedicated ANN solving the two-spiral problem. The topology of the network was adopted from [9], where also an efficient constructive training algorithm called CASCOR was proposed. This so-called "shortcut" topology has four hidden units and 25 links with 25 weights. The activation function is an asymmetric sigmoid function.

The experiment was carried out on two Sun Sparc Stations (one with 128MB RAM and 2 processors and another with 64MB RAM and 1 procesor) connected into the *PVM* environment. The computations were carried out with 6 diferent versions of the PLA-2S with 1, 2, 3, 4, 5 and 6 slave workers, respectively. For each version and each slave initial population

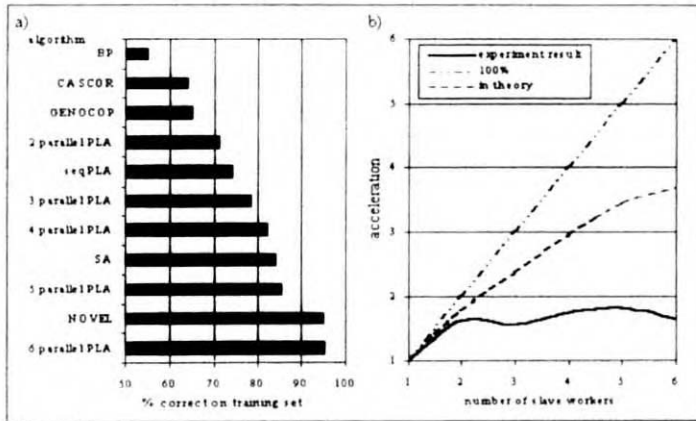


Figure 1: Correct clasifications on training set (a). Acceleration of parallel-PLA versus number of slave workers (b)

Table 1: Comparison of algorithms used for ANN training in two-spiral problem

Approach	Mean time (Sec.)	Mean no. of epochs	MSE
BP	-	20000	-
QP	-	7900	-
CASCOR	-	1700	-
SA	35866	-	-
NOVEL	54000	-	0.04
1-parallel PLA	2580	2815	0.32
3-parallel PLA	809	2118	0.21
6- parallel PLA	1187	4172	0.039

consisted of 200 randomly generated individuals. For PLA-2S with 1, 2 and 3 slave workers the algorithm was run 50 times for each version. In case of 4, 5 and 6 workers computations were carried 15 times for each version. All the reported results are averaged over all runs.

To compare PLA-2S with other approaches the experiment was designed to assure comparability with the results of earlier reported experiments with other algorithms used to train ANN solving the two-spiral problem. In few instances other algorithms were implemented using specifications provided by the respective authors. The reference set of algorithms includes: Cascade-correlation algorithm (CASCOR) [8], NOVEL algorithm [10], standard back propagation algorithm (BP) [4], quickpropagation algorithm (QP), simulated annealing algorithm (SA) [9], GENOCOP [6].

To evaluate the analyzed algorithms three performance measures were used - mean time needed for training, mean number of epochs, and mean squared error (MSE).

Fig. 1a shows percentage of correct classifications on training and test sets as produced by the proposed variants of the PLA-2S and by the reference algorithms. In Table 1 mean times needed to train ANN, number of epochs and the respective MSE are shown.

Fig. 1b is a graph based on the Amdahl's law, showing level of acceleration achieved by introducing parallel versions of the PLA-2S. Actual performance is compared with the ideal case with 100% efficient parallelization and with the theoretical case providing the upper

bound of achievable acceleration. The upper bound has been calculated assuming that 8% of time needed by each slave covers communication delays and has to be executed sequentially.

4 Results Analysis

ANN solving the two-spiral problem, trained by the PLA-2S, demonstrates a very good performance in terms of correct classifications as compared with networks trained using other approaches. The 6-parallel PLA-2S outperforms all other training algorithms producing slightly better results than the second best (NOVEL) and significantly better than all the remaining algorithms. This translates into 0.039 MSE as compared with 0.04 MSE produced by NOVEL. It is worth noting that other versions of the PLA-2S perform also very well in terms of number of the correct classifications and mean squared errors.

Quality analysis of the ANN trained using the discussed algorithms need to be supplemented by evaluation of their time performance. This can be measured directly in terms of computational time needed or indirectly in terms of number of epochs needed to train the respective network. All versions of the PLA-2S clearly outperform all other algorithms in both respects. The only exception here is CASCOR algorithm, which requires similar number of epochs. It should be however noted that the quality of results produced by ANN trained using CASCOR is much lower than in case of training based on PLA-2S.

The acceleration resulting from parallelization of the PLA-2S has proven to be considerably slower than the upper bound represented by the theoretical curve. This difference can be attributed to process communication requirements and diversified environment used in the experiment. Yet, the idea of parallel population learning algorithm is worth further investigations and the increased efficiency of the parallel versions observed during the experiment is a promising factor.

5 Conclusion

Computational experiment carried out shows that the population-learning algorithm is a suitable tool to be considered for training artificial neural networks. For the two-spiral problem PLA-2S proves to be an effective tool from the point of view of both - quality of results and computational time requirements. Computational experiments have shown that for this class of problems even quite basic and simple population learning algorithms can produce remarkably competitive results as compared to other approaches. Further research should concentrate on increasing efficiency of parallelization and on finding better and quicker learning/improvement procedures, as well as on testing the approach using a variety of benchmark problems and neural networks.

References

- [1] Duch, W., Korczak, J.: Optimization and Global Minimization Methods Suitable for Neural Network. *Neural Computing Surveys* 2 (1998), <http://www.icsi.berkeley.edu/~jagopta/CNS>
- [2] Hertz, J., Krogh, A. Palmer, R.G.: *Wstęp do Teorii Obliczeń Neuronowych*. WNT, Warszawa (1995) (in Polish)
- [3] Jędrzejowicz, P.: Social Learning Algorithm as a Tool for Solving Some Difficult Scheduling Problems. *Foundation of Computing and Decision Sciences* 24 (1999) 51-66

- [4] Kevin, J., Witbrock, L., Witbrock, M.J.: Learning to Tell Two Spirals Adapt. in Proceedings of the 1988 Connectionist Models Summer School, Morgan Kaufmann (1998)
- [5] Maniezo, V.: Genetic Evolution of the Topology and Weight Distribution of Neural Networks. IEEE Transactions on Neural Networks 1(5) (1994) 369-53
- [6] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer, Berlin (1996)
- [7] Rutkowska, D., Piliski, M., Rutkowski, L.: Sieci Nauronowe, Algorytmy Genetyczne i Systemy Rozmyte. Wydawnictwo Naukowe PWN, Warszawa (1996) (in Polish)
- [8] Fahlman, S.E., Lebiere, C.: The Cascade-Corelation Learning Architecture. In Touretzky (ed.) Advances in Neural Information Processing Systems 2, Morgan Kaufmann (1990)
- [9] Wah, B.W., Minglun Qian: Constrained Formulations for Neural Network Training and Their Applications to Solve the Two-Spiral Problem. Proc. of Fifth Int'l Conference on Computer Science and Informatics, ICIS, 1 (2000) 598-601
- [10] Yi Shang, Wah, B.W.: A Global Optimization Method for Neural Network Training. Conference of Neural Networks, IEEE Computer 29 (1996) 45-54

Markovian Architectural Bias of Recurrent Neural Networks

Peter Tiňo, Michal Čerňanský, Lubica Beňušková

Neural Computing Research Group,

Aston University, Birmingham, B4 7ET, UK

p.tino@aston.ac.uk, <http://www.ncrg.aston.ac.uk/~tinop>

Department of Computer Science and Engineering FEI,

Slovak Technical University, Ilkovičova 3, 812 19 Bratislava 1, Slovakia

cernans@dcs.elf.stuba.sk, <http://www.dcs.elf.stuba.sk/~cernans>

Institute of Informatics FMFI,

Comenius University, Mlynská dolina, 842 48 Bratislava 4, Slovakia

benus@ii.fmph.uniba.sk, <http://www.ii.fmph.uniba.sk/~benus>

Abstract. Several studies in the cognitive science community (see [1–3]) reported that when training recurrent neural networks (RNNs) to process language structures, activations of recurrent units display a considerable amount of structural differentiation even *prior to learning*. Following [3], we refer to this phenomenon as the *architectural bias of RNNs*. In this paper we show that when initialized with small weights, RNNs produce recurrent activations that cluster according to a Markovian strategy: i.e. subsequences sharing a long common suffix are represented as points in a dense cluster. Prediction states that naturally arise in such networks loosely correspond to states of variable memory length Markov models (VLMM). After training RNNs, the main effort in related studies is concentrated on analyzing activation patterns in the recurrent layer. Given a (trained, or non-trained) RNN and a training stream, we test usefulness of recurrent representations by constructing predictive models, called *neural prediction machines* (NPM), that directly employ state-space dynamics of the network. Based on comparison with the so-called *fractal prediction machines* (FPMs), we demonstrate that NPMs of non-trained RNNs are closely related to VLMMs. Reports on learning and processing linguistic structures using RNNs that concentrate on state-space representations in trained networks should, prior to making any conclusion about the importance of detected recurrent representations, compare RNN results with those of NPMs extracted from untrained nets and/or Markovian models.

Keywords: RNN, Architectural bias, VLMM

1 Introduction

There is a considerable amount of literature devoted to connectionist processing of complex language structures. One of the main driving forces behind such studies has been formulating models of human performance in processing linguistic patterns of various complexity (e.g. [3]). Also, the analysis of state space trajectories in recurrent neural networks (RNNs) has provided new insights into the types of processes which may account for the ability of

learning devices to acquire and represent language, without appealing to traditional linguistic concepts [4, 5]. To gain an additional insight into what the networks have learned, trained networks were sometimes used as generators by transforming their outputs to “probabilities”¹ of possible sentence continuations. One of these possible continuations is then chosen stochastically and fed back as the next input to the network [3, 6].

Yet, several researches issued a note of caution: When training RNNs to process language structures, activations of recurrent units display a considerable amount of structural differentiation even *prior to learning*. We refer to this phenomenon as the *architectural bias of RNNs* [3]. In this paper we provide an explanation of this phenomenon and issue an additional note of caution that the link between the quality of trained RNNs as language generators and the complexity of dynamical state-space scenarios inside the networks is less straightforward than might have been previously thought.

2 Neural Prediction Machines

We work with Elman-type 1st-order randomly-initialized recurrent networks trained using RTRL. The networks consist of 1 input layer, $\mathbf{I}^{(t)}$, and 1 output layer $\mathbf{O}^{(t)}$, each with A units corresponding to the symbols from $\mathcal{A} = \{1, 2, \dots, A\}$ appearing in the training and test sequences. There are 2 hidden layers, $\mathbf{R}^{(t)}$ and $\mathbf{H}^{(t)}$, and 1 context layer, $\mathbf{C}^{(t)} = \mathbf{R}^{(t-1)}$, connected through a unit time delay to the first (recurrent) hidden layer (Fig. 1(f)). The second (non-recurrent) hidden layer, $\mathbf{H}^{(t)}$, was used to increase the flexibility of the maps between the hidden representations in the recurrent portion, $\mathbf{R}^{(t)}$, and the activations at the output layer, $\mathbf{O}^{(t)}$. A logistic sigmoid activation function was used, the learning rate and momentum were set to 0.05, and the training sequence was presented at the rate of one symbol per clock tick. The networks were trained to predict the next item in a sequence given the previous context [3, 7]. At the beginning of each training epoch the network is re-set with the same initial state $\mathbf{C}^{(1)}$. The initial state is randomly chosen prior to the training session.

After training RNNs on linguistic data, the main effort is often concentrated on analyzing the resulting activation patterns in the recurrent layer $\mathbf{R}^{(t)}$. We make explicit use of the induced state-space trajectories in RNNs by formulating prediction models, called *neural prediction machines* (NPM), that inherit state-space dynamics from RNNs. They differ from RNNs only in that they ignore the network output mapping $\mathbf{R}^{(t)} \rightarrow \mathbf{H}^{(t)} \rightarrow \mathbf{O}^{(t)}$, and instead, based on the network dynamics $[\mathbf{I}^{(t)}, \mathbf{C}^{(t)}] \rightarrow \mathbf{R}^{(t)}$, use their own predictive probabilities $P_{NPM}(a|\mathbf{R}^{(t)})$, $a \in \mathcal{A}$.

While the RNN output probabilities are determined, using the map $\mathbf{R}^{(t)} \rightarrow \mathbf{H}^{(t)} \rightarrow \mathbf{O}^{(t)}$, at the network output

$$P_{RNN}(a|\mathbf{R}^{(t)}) = \frac{O_a^{(t)}}{\sum_{b=1}^A O_b^{(t)}}, \quad (1)$$

calculation of the corresponding NPM predictive probabilities involves estimation of the relative frequencies of symbols, conditional on RNN state $\mathbf{R}^{(t)}$. Instead of the map $\mathbf{R}^{(t)} \rightarrow \mathbf{H}^{(t)} \rightarrow \mathbf{O}^{(t)} \rightarrow P_{RNN}(\cdot|\mathbf{R}^{(t)})$, we have a piece-wise constant map, $\mathbf{R}^{(t)} \rightarrow P_{NPM}(\cdot|\mathbf{R}^{(t)})$, estimated on the training sequence. Regions of constant probabilities are determined by vector quantizing recurrent activations that result from driving the network (weights are fixed)

¹By normalizing the sum of outputs to 1.

with the training sequence. In the following, we give a more rigorous description of the NPM construction.

1. Given a training sequence $S = s_1 s_2 \dots s_N$ over an alphabet $\mathcal{A} = \{1, 2, \dots, A\}$, first re-set the network to the initial state $\mathbf{C}^{(1)}$, and then, while driving the network with the sequence S , collect the recurrent layer activation vectors in the set $\Omega = \{\mathbf{R}^{(t)} \mid 1 \leq t \leq N\}$.
2. Run a vector quantizer with M codebook vectors $\mathbf{B}_1, \dots, \mathbf{B}_M$, on the set Ω (we used K-means clustering). Vector quantization partitions the space of recurrent activations into M regions V_1, \dots, V_M , that are Voronoi compartments of the codebook vectors $\mathbf{B}_1, \dots, \mathbf{B}_M$,

$$V_i = \{\mathbf{R} \mid d(\mathbf{R}, \mathbf{B}_i) = \min_j d(\mathbf{R}, \mathbf{B}_j)\},$$

where $d(\cdot, \cdot)$ is the Euclidean distance. All points in V_i are allocated² to the codebook vector \mathbf{B}_i .

3. Re-set the network again with the initial state $\mathbf{C}^{(1)}$.
4. Set the [codebook-vector, next-symbol] counters $N(i, a)$, $i = 1, \dots, M$, $a = 1, \dots, A$, to zero.
5. Drive the network once more with the training sequence S .
For $1 \leq t \leq N$, if $\mathbf{R}^{(t)} \in V_i$, and the next symbol s_{t+1} is a , increment the counter $N(i, a)$ by one.
6. With each codebook vector (or, equivalently, quantization compartment) V_1, \dots, V_M , associate the next symbol probabilities

$$P(a \mid V_i) = \frac{N(i, a)}{\sum_{b \in \mathcal{A}} N(i, b)}, \quad a \in \mathcal{A}. \quad (2)$$

Given a sequence u_1, u_2, \dots, u_L over the alphabet \mathcal{A} , the NPM makes a prediction about the next symbol u_{L+1} as follows:

1. Re-set the network with the initial state $\mathbf{C}^{(1)}$.
2. Drive the network with the sequence u_1, u_2, \dots, u_L , i.e. for $1 \leq t \leq L$, recursively compute the vectors $\mathbf{R}^{(t)}$.
3. The next symbol distribution given by NPM is
 $P_{NPM}(u_{L+1} \mid \mathbf{R}^{(L)}) = P(u_{L+1} \mid V_i)$, provided $\mathbf{R}^{(L)} \in V_i$.

Note, that NPM is *not* a finite state predictor. The dynamics of NPM is completely the same as that of the associated RNN. NPM makes use of spatial representations of symbol histories encoded by the RNN. The history of symbols seen by the network up to time t is represented by the vector of recurrent activations $\mathbf{R}^{(t)}$. Quantization of RNN state space merely helps us to estimate, on a (finite) training sequence, the state-conditional predictive probabilities $P(\cdot \mid \mathbf{R}^{(t)})$ by a piece-wise constant map.

²Ties as events of measure zero (points land on the border between the compartments) are broken according to index order.

2.1 Fractal Prediction Machines

In [8] we introduced a special type of NPM, called *fractal prediction machine* (FPM). FPMs are constructed from RNNs with dynamics of an affine iterative function system (IFS). Assuming the network input at time t is $a \in \mathcal{A} = \{1, 2, \dots, A\}$, the dynamics is defined by

$$\mathbf{R}^{(t)} = \frac{1}{2}(\mathbf{C}^{(t)} + \mathbf{T}_a), \quad (3)$$

where $\mathbf{T}_1, \dots, \mathbf{T}_A, \mathbf{T}_a \neq \mathbf{T}_b$, for $a \neq b$, are symbol-related vertices of the hypercube $[0, 1]^D$, and the dimension D is no less than $\log_2 A$. Note that such non-autonomous dynamics³ is driven by A attractive fixed points $\mathbf{T}_1, \dots, \mathbf{T}_A$, corresponding to symbols from the alphabet \mathcal{A} .

We showed that FPMs are closely related to variable memory length Markov models (VLMM) [9], which are Markov models of flexible memory depth, depending on the prediction context. Roughly speaking, VLMMs can utilize deep memory only when it is really needed, thus avoiding explosion in the number of free parameters of the fixed-order Markov models. The IFS dynamics, Eq. (3), driven by attractive fixed points associated with symbols from \mathcal{A} , transforms subsequences appearing in the training sequence into a set of points in Euclidean space, such that points corresponding to blocks sharing a long common suffix are mapped close to each other. Vector quantization on such a set partitions histories of seen symbols into several classes dominated by common suffixes. Quantization centers of FPMs play the role of predictive contexts in VLMMs. FPMs and VLMMs achieved comparable performances across a collection of symbolic sequences from a wide range of complexity and memory structures (see [8]).

3 Processing recursive structures

In this section we train RNNs and construct NPMs on three data sets of the type used by Christiansen and Chater to assess connectionist modeling of recursion in human linguistic performance [3]. They trained simple recurrent network [7] on the next-symbol prediction and also assessed the trained networks as sequence generators.

Each language used in [3] involves one of three complex recursions taken from Chomsky [10], interleaved with right-branching recursions (RBR). The latter is generated by a simple iterative process to obtain constructions like: $P_N P_V S_N S_V$, where P stands for plural, S for singular, N for noun and V for verb category. Example: “girls like the boy that runs”. The three complex recursions are:

(1) Counting recursion (CR): $\{\}, NV, NNVV, NNNVVV, \dots$, while ignoring singulars and plurals.

(2) Center-embedding recursion (CER): $\{\}, \dots, S_N P_N P_V S_V, P_N S_N S_V P_V, \dots$. Example: “the boy girls like runs”.

(3) Identity (cross-dependency) recursion (IR): $\{\}, \dots, S_N P_N S_V P_V, P_N S_N P_V S_V, \dots$. Example: “the boy girls runs like”.

³Strictly speaking, FPMs were in [8] constructed as finite-memory sources, where the IFS dynamics was driven only by the last L recently seen symbols. Typically, $L = 20$. However, the distance between recurrent activations resulting from 2 sequences that share a common suffix of length L is less than $2^{-L}\sqrt{D}$ (see [11]). So, for long memory depths L , the dynamics of the FPM introduced above is virtually the same as that of the FPM described in [8].

Thus, our three benchmark recursive languages were: CRandRBR, CERandRBR, IRandRBR. Each language had a 16-word vocabulary with 4 words from each category, i.e. 4 singular nouns, 4 singular verbs, 4 plural nouns and 4 plural verbs. The RNN had 17 input and output units, where each unit represented one word or the end of sentence mark. There were 10 hidden and 10 recurrent neurons. The networks were trained in 10 training runs starting from different random weight initializations (from $[-0.5, 0.5]$). The training set of each language consisted of 5000 sentences and the test set of 500 novel sentences. One half of each set was comprised of RBR constructions and another half of appropriate complex recursions. Depths of embedding ranged from 0 to 3, with the following distributions: depth 0 – 15 %, depth 1 – 27.5 %, depth 2 – 7 %, depth 3 – 0.5 % (together 50 %). The mean sentence length was approximately 4.7 words.

In **Fig. 1(a)–(e)** we show the mean (across 10 training runs) normalized (per symbol) negative log likelihoods (NNL) achieved on the test set by FPMs, RNNs and the corresponding NPMs. In the 2-D plots, we also show the corresponding standard deviations. Standard deviations in the 3-D plots are not shown, but generally are less than 5 % of the mean value.

The effect of architectural bias is clearly visible. This is confirmed by comparing NPM performances (for 0 training epochs) in **Fig. 1(c)–(e)** with the FPM performance in **Fig. 1(a)**, effectively implementing VLMMs [8]. About 40 codebook vectors in NPMs are sufficient to make full use of the associated RNNs dynamics. Moreover, NPMs corresponding to non-trained networks achieved NNL comparable with NNL given by RNNs after 10 epochs of training (**Fig. 1(b)**). What is the explanation for this, rather surprising, behavior? The dynamics of RNNs initialized with small weights is trivial. With a fixed input, it is completely dominated by a single attractive fixed point close to the center of the recurrent activation space (RNN state space) [12]. Different input codes of symbols $1, 2, \dots, A$, form \mathcal{A} correspond to A different attractive fixed points in the RNN state space. Hence, when driving the network with an input sequence over \mathcal{A} , the network codes histories of seen symbols in its recurrent layer in the same Markovian manner as FPMs do (subsection 2.1). NPMs of non-trained RNNs, initialized with small weights, are much like FPMs. Before the training, RNN outputs are driven by randomly initialized weights in the output mapping $R^{(t)} \rightarrow H^{(t)} \rightarrow O^{(t)}$ and it takes some time for the mapping to adjust to the statistics of the training sequence. Also, since initially the A attractive fixed points lie in a close neighborhood of the center of the state space, it may be difficult for the RNN to adjust the smooth output map to the dynamics that takes place in a rather tiny portion of the state space. But, the “useful” dynamics is already there, even prior to the training (!), and NPMs are able to make full use of it, since NPM predictive probabilities are computed as piece-wise constant maps based on (scale-free) vector quantization of RNN state space.

4 Conclusion

We have offered an explanation of the phenomenon known to the cognitive science community as the architectural bias in recurrent neural networks (RNNs). RNNs initialized with small weights are biased towards the class of Markov models known as variable memory length Markov models. We constructed predictive models, called neural prediction machines (NPM), that share the state-space dynamics of RNNs, but are not dependent on the RNN output map. Using NPMs we were able to test the usefulness of state space representations in RNNs for building probabilistic models of linguistic data. Experiments on recursion data

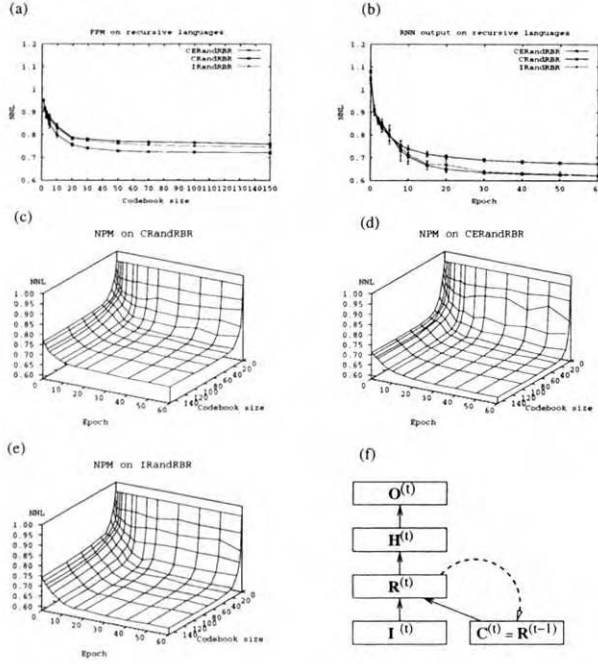


Figure 1: Normalized negative log likelihoods (NNL) achieved on Christiansen and Chater recursion data sets by (a) FPMs, (b) output of RNNs and (c)–(e) NPMs. (f) RNN.

of Christiansen and Chater confirmed our Markovian architectural bias hypothesis. Although not reported here, we obtained very similar results on a minimal-pair linguistic data set based on the University of Pennsylvania ‘Brown’ corpus.

Acknowledgements: Supported by VEGA 1/9046/02.

References

- [1] Kolen, J.F.: The origin of clusters in recurrent neural network state space. *Proc. 16th Annual Conf. Cognitive Sci. Soc.*, Lawrence Erlbaum Assoc., Hillsdale, NJ (1994) 508–513
- [2] Chater, N., Conkey P.: Finding linguistic structure with recurrent neural networks. *Proc. 14th Annual Meet. Cognitive Sci. Soc.*, Lawrence Erlbaum Assoc., Hillsdale (1992) 402–407
- [3] Christiansen, M.H., Chater, N.: Toward a connectionist model of recursion in human linguistic performance. *Cognitive Science*, **23** (1999) 157–205
- [4] Parfitt, S.: Aspects of anaphora resolution in artificial neural networks: Implications for nativism. *PhD thesis*, Imperial College, London (1997)
- [5] Servan-Schreiber, D. et al.: Graded state machines: The representation of temporal contingencies in Simple Recurrent Networks. In *Advances in Neural Information Processing Systems*. Morgan-Kaufmann (1989) 643–652
- [6] Mozer, M., Soukup T.: Connectionist music composition based on melodic and stylistic constraints. In *Advances in Neural Information Processing Systems 3*. Morgan-Kaufmann (1991) 789–796
- [7] Elman, J.L.: Finding structure in time. *Cognitive Science* **14** (1990) 179–211
- [8] Tiño, P., Dorffner, G.: Predicting the future of discrete sequences from fractal representations of the past. *Machine Learning* **45** (2001) 187–218
- [9] Ron, D., Singer, Y., Tishby, N.: The power of amnesia. *Machine Learning* **25** (1996) 138–153
- [10] Chomsky, N.: *Syntactic Structures*. Mouton, The Hague (1957)
- [11] Tiño, P.: Spatial representation of symbolic sequences through iterative function systems. *IEEE Tran. Syst., Man, and Cyber. Part A: Systems and Humans* **10** (1999) 284–302
- [12] Tiño, P., Horne, B.G., Giles, C.L.: Attractive periodic sets in discrete time recurrent networks (with emphasis on fixed point stability and bifurcations in two-neuron networks). *Neural Computation* **13** (2001) 1379–1414

SOFM Training Speedup

Marcel JIŘINA

Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University
Prague, Technická 2, 166 27 Prague 6, Czech Republic,
jirina@labe.felk.cvut.cz

Abstract. In this paper we introduce a new approach to pre-processing (initial setting) of weight vectors and thus a speed-up of the well-known SOFM neural network. The idea of the method consists in a consequent use of two partial methods: *Init1* and *Filter*. The *Init1* method orders nodes in a lattice with randomly set initial weight vectors. Such a rough ordering is consequently “smoothed” by the *Filter* method that filters values of weights in each lattice direction (along lattice “fibers”). The final result of this pre-setting of weight vectors is a lattice that is visually close to the final state of the lattice. Hence, the training (or now better called tuning) can run for extremely short time. To demonstrate the suggested method we show its abilities on a representative example.

Keywords: SOM, SOFM, Kohonen’s Neural Network, Training Speedup, Initial Setting of Weights

1. Introduction

The idea of training speedup of the self-organizing feature map neural network (SOFM) is based on proper pre-processing (initial setting) of weight vectors in the SOFM neural network. The better ordering of the lattice, i.e. the greater similarity of the initial lattice with the trained lattice, the faster training of the SOFM.

The initial weights between the input and the competitive layer are usually set to “random values” in a wide spectrum of recent publications [1], [2], [3], [6], [7], [8], [9]. Sometimes, a subset of Q training patterns is directly taken from the training set in the random order, where Q is the number of nodes in the SOFM, and the taken patterns are in some way assigned to all the weight vectors [1], [7], [8]. It allows to roughly estimate the density of the pattern space. Another way how to set the weight vectors is to set them all on the same values [1]. Unfortunately, such an initialization does not respect the mutual relations among the nodes at the beginning of the training and thus the mutual relations have to be created during the training.

The idea of the presented method consists in a consequent use of two partial methods: *Init1* and *Filter*. The *Init1* method orders nodes in a lattice with randomly set initial weight vectors. Such a rough ordering is consequently “smoothed” by the *Filter* method that filters values of weights in each lattice direction (along lattice “fibers”). The final result of this pre-setting of weight vectors is a lattice that is visually close to the final state of the lattice. Hence, the training (or now better called tuning) can run for an extremely short time. To demonstrate the suggested method we show its abilities on a representative example.

2. Initial Weight Vectors Setting and Filtering

In the *Init1* method [4], [5] we first of all take patterns from the training set in a random manner. The number of those patterns directly corresponds to the total number of nodes or

neurons in the lattice. Afterwards we assign these patterns to the weight vectors. In this way we have distributed the weight vectors over the pattern space to roughly approximate or reflect the distribution of the training patterns in the pattern space. However, the weight vectors determine the node positions, and therefore the lattice composed of these nodes will be disordered. Furthermore, we describe a procedure for proper ordering of the weight vectors (nodes in the lattice) lattice in the two-dimensional (2D) pattern space and filtering such created lattice. The aim is to achieve as good as possible estimation of the final lattice. The ordering of higher dimensional lattice in higher dimensional space is similar. If the space dimension is higher than the lattice dimension then an orthogonal projection of the lattice is used.

The ordering of a lattice in the two-dimensional (2D) pattern space runs this way. Assume an arbitrary weight vectors (nodes) distribution over the pattern space and the 2D lattice has a rectangular form with $m_1 \times m_2$ nodes. In the first step of the *Init1* method we take from the training set so many patterns, as there are nodes in the lattice, i.e. $m_1 \times m_2$. In Figure 1 is an example of a 2D lattice of 4×4 nodes in a 2D pattern space.

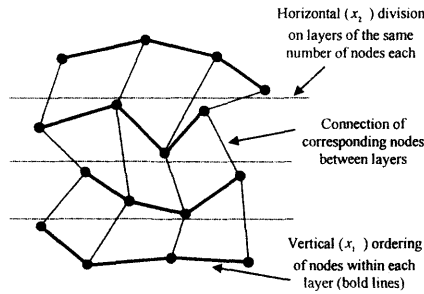


Figure 1. The principle of the ordering process.

The second step of the *Init1* method, i.e. the ordering, can be divided into two sub-steps. In the first step we order all weight vectors according to their x_2 -coordinates. Then we split the ordered weight vectors onto each m_1 groups which contains m_2 weights. On each group we apply the ordering algorithm for a 1D lattice, i.e. a normal sorting algorithm, to obtain ordered weight vectors along their x_1 -coordinates. In other words, we have transformed the 2D lattice on m_1 1D lattices by dividing the original lattice into layers. After processing all m_1 individual groups or layers, the lattice will already get near to the final lattice we are estimating. (Of course, it is possible to exchange the x_1 and x_2 -coordinates and thus turn the lattice ordering.)

The Filter method follows up a previously ordered lattice e.g. by the *Init1* method and further improves the lattice shape. Note that the *Filter* method can be applied to any MD lattice in any ND pattern space, because the method is based on fiber processing, where a fiber is an 1D sub-lattice in one given lattice direction.

The basic idea of the proposed Filter method is based on "smoothing" each fiber of a lattice, i.e. on suppressing extreme values of node coordinates. The smoothing of a fiber is in fact a filtering of node coordinates of the fiber. In the Filter method we can use any filter, but here we have restricted ourselves only to the most widespread averaging filter. For the smoothing or filtering we use a window, which step-by-step moves or slides by one node over each fiber in the lattice and we calculate an average of node coordinates inside the actual window. We will call this window a sliding window. By the obtained averaged coordinate values we replace the coordinates of the central node in the window, see

Figure 2. We have to average all coordinates of a given node except the node coordinate, which determines the fiber orientation in the lattice. In the Figure 2 the x_2 -coordinates remained unchanged.

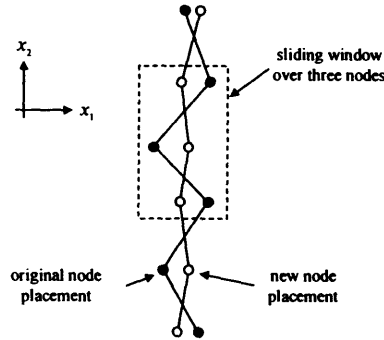


Figure 2. Fibers smoothing using the *Filter* method with an averaging filter over three fiber nodes.

3. Experiments and Results

To demonstrate the influence of the *Init1* and *Filter* methods on the quality of node ordering we show a representative example for a 2D lattice. For all demonstrations based on the *Filter* method a sliding window with 3×3 nodes and a lattice with 15×15 nodes was used. We can see the results in the Figure 3, the *Filter* method expressively improves the *Init1* node ordering in the lattice. Moreover, this method reflects the smoothness of fibers set by the basic SOFM.

4. Setting of Other Parameters of the SOFM after the Weight Pre-processing

After the initial setting of weight vectors (using the *Init1* and/or *Filter* method) we do not start the training process from the beginning because the lattice is already roughly unfolded over the pattern space. It means that all parameters of the training, i.e. the number of training steps, gain term and neighborhood sizes, have to be already set to lower values in comparison with their original initial values in the basic training algorithm for the SOFM. If the gain term stays large (near to one) then it will cause great node shifts and thus it will destroy the preset weight vectors, because it will return the lattice nodes from their correct local areas to the wrong ones. A similar problem will arise with large initial neighborhood sizes around the nodes. The learning of the SOFM with an initially ordered lattice can be denoted it as up-to-training or simply tuning.

On the basis of experimental observations all the parameters, i.e. the number of training steps, the gain term and the neighborhood sizes can fluctuate from $\frac{1}{10}$ to $\frac{1}{5}$ of their original initial values.

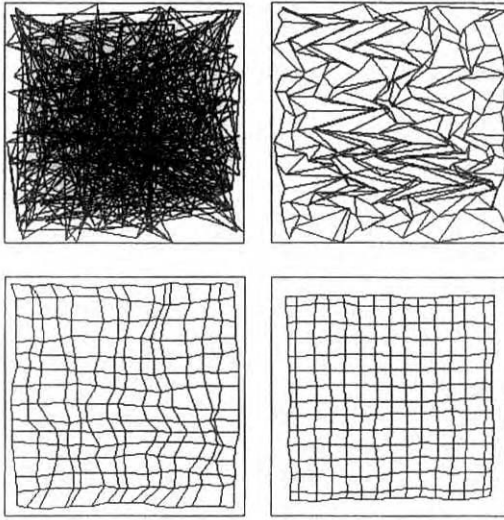


Figure 3. The lattice with randomly selected initial weights, the lattice after processing by the *Init1* method, the lattice after processing by the *Filter* method, and the final lattice after training (tuning). The square shaped pattern space has been used.

5. Conclusion

In this paper we have presented a new method allowing a better setting of initial weights in the SOFM. The basic idea of proper weight pre-processing follows from an ordering of lattice nodes in the lattice (method *Init1*) and consequently filtering the node weight vectors in the lattice using a filter, typically an averaging one (method *Filter*). The method is applicable for an arbitrary distribution of patterns and pattern space dimension and form. A representative example in the 2D space has been shown.

6. Novelty of the Suggested Approach

The method orders and filters the lattice to estimate the final ordering of the lattice and therefore it significantly shortens the training process. Experimental results from a range image segmentation [5] have shown that when using the introduced methods the total training times is reduced more than ten times.

References

- [1] Beale, R. and T. Jackson. *Neural Computing: An Introduction*. Adam Hilger, Bristol Philadelphia New York, 1990
- [2] Fausett, L. *Fundamentals of Neural Networks: Architectures, Algorithms and Applications*. Prentice-Hall, Inc., Englewood Cliffs, 1994
- [3] Haykin, S. *Neural Networks. A Comprehensive Foundation*. Macmillan College Publishing Company, Inc., 1994.
- [4] Jirina, M. *Initial Settings of Weights in the Kohonen Maps*. PhD thesis, Czech Technical University Prague, Faculty of Electrical Engineering, 1998
- [5] Jirina, M. *Kohonen's Maps for Range Image Segmentation*. MS thesis, Charles University Prague, Faculty of Mathematics and Physics, 2000
- [6] Kohonen, T. Self-Organized Formation of Topologically Correct Feature Maps. *Biological Cybernetics*, 43:59-69, 1982
- [7] Kohonen, T. The Self-Organizing Map. *Proceedings of the IEEE*, 78(9), 1990

- [8] Kohonen, T. Self-Organizing Maps. Springer-Verlag, Springer Series in Information Sciences, Berlin Heidelberg New York, 1995
- [9] Lippmann, R. P. An Introduction to Computing with Neural Nets. IEEE ASSP Magazine, April 1987

Generalized Forecasting Sigma-Pi Neural Network

Yevgeniy BODYANSKIY¹, Vitaliy KOLODYAZHNIY¹, Nonna KULISHOVA²

*Control Systems Research Laboratory,
Kharkiv National University of Radioelectronics,
14, Lenin Av., Kharkiv, 61166, Ukraine*

¹ {Bodyanskiy, Kolodyazhniy}@ieee.org

² kunonna@mail.ru

Abstract. The problem of time series forecasting using the sigma-pi artificial neural network is considered. This network combines the advantages of the multilayer perceptrons and radial basis function networks. The specific feature of the considered network is the presence of two different types of activation functions: sigmoid and bell-shaped. A learning algorithm for the sigma-pi network is proposed. The algorithm is not based on error backpropagation, and is significantly simplified due to the use of the polynomial approximation of the nonlinear activation functions. Comparison with the multilayer perceptron in the forecasting of the sunspot time series is given.

1. Introduction

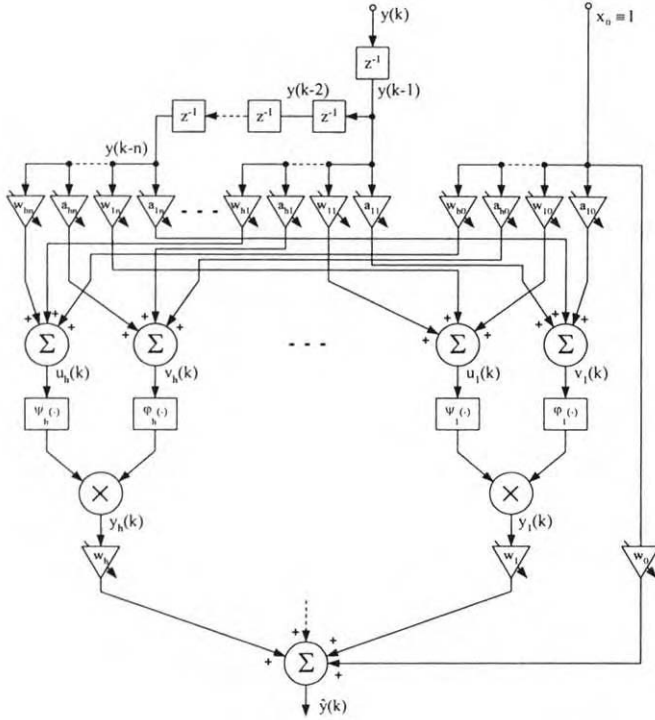
Artificial neural networks (ANN) are widely used in solving many problems of information processing, and, in particular, in the forecasting of various stochastic signals. Both the feedforward [1-4] and recurrent [5-7] multi-layer architectures are popular in these applications, as well as the forecasting radial basis function (RBF) neural networks [8, 9]. The efficiency of the multi-layer networks is explained by their universal approximation properties. However, slow learning, based on the concept of error back-propagation, may complicate the forecasting of the substantially non-stationary processes. As an alternative to the multi-layer networks, the RBF networks can be used. These networks contain only one hidden layer, and the activation functions, in contrast to the sigmoids of the multilayer ANN, are non-monotonic, e.g. the gaussian functions.

The most important advantage of the RBF networks lies in the reduced training time, as compared to the networks trained through the error back-propagation, in conjunction with high approximation accuracy. At the same time, the number of neurons in the hidden layer must be large, and it increases exponentially as the dimensionality of the approximated function grows.

The advantages of both the multilayer and RBF networks are combined in the so-called Σ - Π (sigma-pi) neural networks [10, 11] that have found rather limited application by now.

2. Network Architecture

The architecture of the forecasting Σ - Π ANN is shown in Figure 1.

Figure 1. Architecture of the forecasting Σ - Π neural network

The network consists of delay units, adaptive linear associators (adalines), nonlinear activation function units, and multipliers. The signal of the predicted time series $y(k)$, $k = 0, 1, 2, \dots$ is fed to the input layer of the network, which is comprised of the delay units, forming a $(n+1) \times 1$ vector of delayed measurements $x(k) = (1, y(k-1), y(k-2), \dots, y(k-n))^T = (x_0(k), x_1(k), x_2(k), \dots, x_n(k))^T$, which is then fed to the hidden layer. This layer consists of h neurons. Each neuron contains two summation units with $n+1$ adjustable inputs, two nonlinear activation function units (one sigmoid and one bell-shaped), and a multiplier. The output layer consists of one neuron, similar to an adaptive linear associator with $h+1$ inputs. The network under consideration contains $1 + h(2n+3)$ adjustable synaptic weights altogether. The forecast, provided by the Σ - Π network, is calculated according to the equation

$$\hat{y}(k) = w_0(k) + \sum_{i=1}^h w_i(k) \varphi_i(v_i(k)) \psi_i(u_i(k)), \quad (1)$$

where $v_i(k) = \sum_{j=0}^n a_{ij}(k) x_j(k)$, $u_i(k) = \sum_{j=0}^n w_{ij}(k) x_j(k)$, $i = 1, \dots, h$, $j = 0, \dots, n$,

$w_0(k)$, $w_i(k)$, $w_{ij}(k)$, $a_{ij}(k)$ are the adjustable synaptic weights at the time instant k , $\varphi(\bullet)$ is the bell-shaped activation function of the i -th neuron, $\psi(\bullet)$ is the sigmoid activation function of the same neuron.

Denoting the output signal of the i -th neuron in the hidden layer as $y_i(k) = \varphi_i(v_i(k)) \psi_i(u_i(k))$, we can re-write the equation (1) in the following form:

$$\hat{y}(k) = w_0(k) + \sum_{i=1}^h w_i(k) y_i(k). \quad (2)$$

As it can be readily seen from the equation (2), the signal of the output layer is formed similarly to the output of an RBF network, and the hidden layer possesses the properties of both the perceptrons and RBF networks.

3. Learning Algorithm

Introducing a local learning criterion $E(k) = 1/2 e^2(k)$ (where $e(k) = y(k) - \hat{y}(k)$ is the forecast error) and a vector $(1 + h(2n + 3)) \times 1$ of the tuned synaptic weights $W(k) = (w_0(k), w_1(k), \dots, w_h(k), \dots, w_{hj}(k), \dots, a_{ij}(k), \dots)^T$, we can write the learning algorithm for the Σ - Π network in the following form:

$$\begin{cases} w_0(k+1) = w_0(k) + \mu_0(k)e(k), \\ w_i(k+1) = w_i(k) + \mu_i(k)e(k)y_i(k), \\ w_{ij}(k+1) = w_{ij}(k) + \mu_{ij}(k)e(k)w_i(k)\varphi_i(v_i(k))\frac{d\psi_i(u_i(k))}{du_i(k)}x_j(k), \\ a_{ij}(k+1) = a_{ij}(k) + \mu_{ij}(k)e(k)w_i(k)\psi_i(u_i(k))\frac{d\varphi_i(v_i(k))}{dv_i(k)}x_j(k), \end{cases} \quad (3)$$

where $\mu_{ij}(k)$ are the learning rates.

Computational complexity of the algorithm (3) and the need to compute partial derivatives of the activation functions at each time instant necessitates searching the ways to simplify the learning procedure. It is helpful to note that the sigmoid activation function $\psi(\gamma q)$, defined on the quadrate $-1 \leq q \leq 1$, $-1 < \psi(\gamma q) < 1$, $\gamma > 0$, can be represented, with high accuracy and comparatively small number of expansion terms L , as the polynomial series

$$\begin{aligned} \psi(\gamma q) &= \alpha_0 \gamma q + \alpha_1 \gamma^3 q^3 + \alpha_2 \gamma^5 q^5 + \alpha_3 \gamma^7 q^7 + \dots = \\ &= \sum_{l=0}^L \alpha_l (\gamma q)^{2l+1}, \quad \alpha_{l+1} = \beta \alpha_l, \quad -1 < \beta < 0, \end{aligned} \quad (4)$$

and the first derivative of this expansion

$$\begin{aligned} \frac{\partial \psi(\gamma q)}{\partial q} &= \alpha_0 \gamma + 3\alpha_1 \gamma^3 q^2 + 5\alpha_2 \gamma^5 q^4 + 7\alpha_3 \gamma^7 q^6 + \dots = \\ &= \sum_{l=0}^L \gamma(2l+1)\alpha_l (\gamma q)^{2l} = \sum_{l=0}^L \alpha'_l (\gamma q)^{2l}, \quad \alpha'_l = \gamma(2l+1)\alpha_l \end{aligned} \quad (5)$$

is bell-shaped, and the second derivative

$$\begin{aligned} \frac{\partial^2 \psi(\gamma q)}{\partial q^2} &= 2 \cdot 3\alpha_1 \gamma^3 q + 4 \cdot 5\alpha_2 \gamma^5 q^3 + 6 \cdot 7\alpha_3 \gamma^7 q^5 + \dots = \\ &= \sum_{l=0}^L 2\gamma^2(l+1)(2l+3)\alpha_{l+1} (\gamma q)^{2l+1} = \sum_{l=0}^L \alpha''_l (\gamma q)^{2l+1}, \quad \alpha''_l = 2\gamma^2(l+1)(2l+3)\alpha_{l+1} \end{aligned} \quad (6)$$

is sigmoid again.

The expansion (4)-(6) allows us to unify the components in the equation (2), and we can write the learning algorithm in the form

$$\begin{cases}
 w_0(k+1) = w_0(k) + \mu(k)e(k), \\
 w_i(k+1) = w_i(k) + \mu(k)e(k) \left(\sum_{j=0}^L \alpha_j'(v_j(k))^{2j} \sum_{l=0}^L \alpha_l(u_l(k))^{2l+1} \right), \\
 w_y(k+1) = w_y(k) + \mu(k)e(k) w_i(k) \left(\sum_{j=0}^L \alpha_j'(v_j(k))^{2j} \sum_{l=0}^L \alpha_l(u_l(k))^{2l} x_j(k) \right), \\
 a_{ij}(k+1) = a_{ij}(k) + \mu(k)e(k) w_i(k) \left(\sum_{j=0}^L \alpha_j(u_j(k))^{2j+1} \sum_{l=0}^L \alpha_l'(v_l(k))^{2l+1} \right) x_j(k).
 \end{cases} \quad (7)$$

4. Simulation Results

The proposed network and learning algorithm were tested on the well-known sunspot time series. The results were also compared with those obtained using a perceptron with one hidden layer, containing the hyperbolic tangent activation functions, and a linear output layer. Both the Σ - Π network and the perceptron had 11 inputs for the corresponding delayed values of the time series, and contained 5 neurons in hidden layers.

The time series contained 288 values of the average number of sunspots for the years of 1700-1987, and was divided into the training data set of the first 208 values, and the checking data set of the remaining 80 values. The data were normalized to the range of [-1, 1].

For the approximation of the activation functions in the Σ - Π network, we used 4 terms of the expansion (4)-(6). The coefficients of the polynomials were identified through the least squares fitting.

The Σ - Π network was trained using the proposed algorithm (7) with the learning rates of 0.005. The perceptron was trained on the same training data set, at first using the gradient descent training procedure with the learning rate of 0.005, and then using the Levenberg-Marquardt procedure.

In all the three experiments, the networks were trained for 100 epochs. After the training, the forecasts of the values in the training and checking data sets were calculated.

The forecast precision was estimated using the root mean squared error on the training data set ($RMSE_{TRN}$) and checking data set ($RMSE_{CHK}$). The forecasts of the training data and checking data for the Σ - Π network are shown in Figure 2 and Figure 3, respectively. All the results are summarized in Table 1.

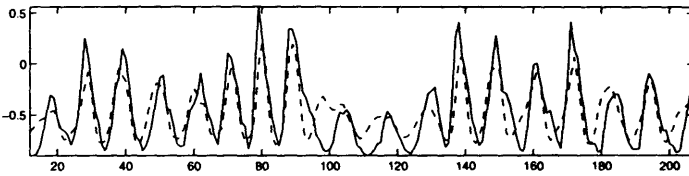


Figure 2. Forecasting results for the Σ - Π network on the training data set: $RMSE_{TRN}=0,2041$

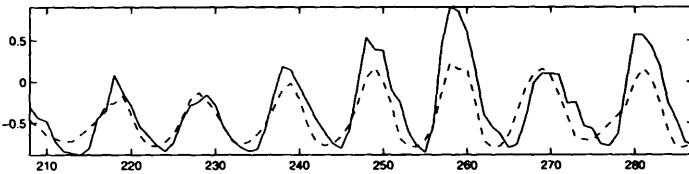


Figure 3. Forecasting results for the Σ - Π network on the checking data set: $RMSE_{CHK}=0,2575$

The Σ - Π network trained with the algorithm (7) considerably outperformed the perceptron trained with the gradient descent procedure, and provided similar forecast precision on the checking data set as the perceptron trained with the Levenberg-Marquardt procedure (which is known to be one of the most effective training procedures for the ANN, and is much more complex computationally).

Network/Training procedure	RMSE _{TRN}	RMSE _{CHK}
Σ - Π /Proposed	0.2041	0.2575
Perceptron/Gradient descent	0.4195	0.5718
Perceptron/Levenberg-Marquardt	0.0784	0.2563

Table 1. Summary of simulation results

5. Conclusion

Simulation results confirm high forecast precision, provided by the considered non-conventional Σ - Π ANN. High performance of the network is achieved due to the use of a new learning algorithm, which is quite simple computationally and does not employ the error back-propagation.

Further research is necessary to investigate the dependence of the network performance on the learning rates in the proposed algorithm, and to determine the optimal number of expansion terms in the polynomial approximation of the activation functions. It is expected that the considered Σ - Π networks can be used as an effective tool for the forecasting of various, including the non-stationary, time series.

References

- [1] Pham, D.T., Liu, X.: *Neural Networks for Identification, Prediction and Control*. Springer-Verlag, London (1995)
- [2] Masters, T.: *Neural, Novel & Hybrid Algorithms for Time-Series Prediction*. John Wiley & Sons, Inc., New York (1995)
- [3] Saxen, H.: Nonlinear Time Series Analysis by Neural Network – A Case Study. *Int. J. Neural Systems* 7 (1996) 195-201.
- [4] Conway, A.J., Macpherson, K.P., Brown, J.C.: Delayed Time Series Prediction with Neural Networks. *Neurocomputing* 18 (1998) 81-89
- [5] Connor, J.T., Martin, R.D., Atlas, L.E.: Recurrent Neural Networks and Robust Time Series Prediction. *IEEE Trans. on Neural Networks* 5 (1994) 240-254
- [6] Aussem, A., Murtagh, F., Sarazin, M.: Dynamical Recurrent Neural Networks – Towards Environmental Time Series Prediction. *Int. J. Neural Systems* 6 (1995) 145-170.
- [7] Madhavan, P.G.: A New Recurrent Neural Network Learning Algorithm for Time Series Prediction. *J. Intelligent Systems* 7 (1997) 103-116.
- [8] Chang, E.S., Chen, S., Mulgrew, B.: Gradient Radial Basis Neural Networks for Nonlinear Time Series Prediction. *IEEE Trans. on Neural Networks* 7 (1996) 190-194.
- [9] Billings, S.A., Hong, X.: Dual-orthogonal Radial Basis Function Networks for Nonlinear Time Series Prediction. *Neural Networks* 11 (1998) 479-493.
- [10] Chichocki, A., Unbehauen, R.: *Neural Networks for Optimization and Signal Processing*. Teubner, Stuttgart (1993)
- [11] Bothe, H.-H.: *Neuro-Fuzzy Methoden. Einfuehrung in Theorie und Anwendungen*. Springer-Verlag, Berlin (1998)

Human Centered Intelligent Robots Using "Ontological Neural Network"

Toru YAMAGUCHI **, Eri SATO*, Sei WATANABE*

/Department of Electronic Systems Engineering, Tokyo Metropolitan Institute of
Technology 6-6, Asahigaoka, Hino, Tokyo, 191-0065, JAPAN*

*** PRESTO, Japan Science and Technology Corporation (JST)*

Abstract. In recent years, the necessity for care has been increasing. But care for workers has decreased in number. So "Human Centered Intelligent Robots" attracts attention. Welfare support robots must understand their role among themselves, and robots share information to each other. In this paper, agents use Soft DNA (Soft computing oriented Data driven functional scheduling Architecture) and Ontological Neural Network.

Keywords: Ontology, Intelligent robots

1. Introduction

Now, Japan has become a society composed largely of elderly people, and the number of births is declining. The necessity for care has been increasing. But care workers has decreased in number. So "Human Centered Intelligent Robots" attract out attention. Human Centered Intelligent Robots support elderly people and a handicapped daily life. Welfare support robots must understand their role among themselves, and robots share information to each other. Agents use Soft DNA (Soft computing oriented Data driven functional scheduling Architecture) and Ontological Neural Network. In this paper, plural robots using Ontological Neural Network and soft DNA smooth support to human are described.

2. Intelligence and control structure

A robot's control mechanism assumes a hierarchy intelligence model. The hierarchy intelligence model is based upon the hierarchy model of Rasmussen. The hierarchy model of Rasmussen is known as an effective model to construct an intelligent model. We extend the hierarchy model of Rasmussen, group intelligence treating an abstract phenomenon, the higher layer treating a concrete phenomenon, the lower layer treating the local feedback control. It has built a robot's intelligent structure described above (Fig.1).

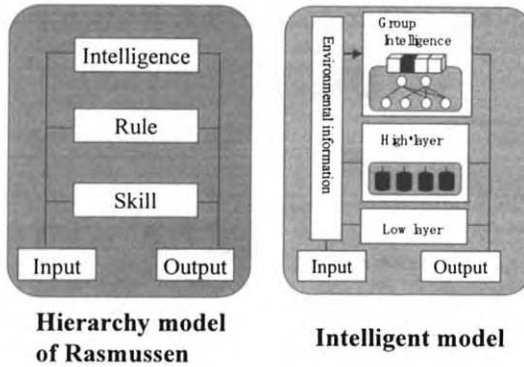


Figure 1: Rasmussen and Intelligent model

3. Ontological Neural Network

3.1 Soft DNA

Soft DNA (Soft computing oriented Data driven functional scheduling Architecture) aims to imitate the idea of the developmental process, such as the body plans in the actual life based on biological DNA (Deoxyribo-Nucleic-Acid).

In biological DNA, the genes called “Homeo box genes” dynamically control the body development of an individual in the actual life based on the concentrations of proteins in cells. The control architecture called “Soft DNA” dynamically controls the developmental information in order to activate dynamic cooperation. Biological DNA has sets of genes, and the genes are related to the body such as the head, chest, abdomen, and tail. These all sets of genes are called a homeo box.

Similarly, Soft DNA has boxes of intelligence (made by soft computing, i.e., associative memories, neural networks, fuzzy logic, chaos, and so on) that are related to various environments, a suitable box of intelligence is developed according to the environmental information available as shown in Fig.2.

Soft DNA can form a role numbers of times and can realize the structure of each knowledge for organizing.

3.2 Ontological Neural Network

A common concept like a word for humans is needed for a conversation between robots. Therefore, we construct a concept based on the conversation. As for ontology, an abstract general idea structure was defined. The ontology is expressed by the relations with more than one concrete case (instance).

Robots realize the conversation by Ontological Neural Network. (Fig.3).

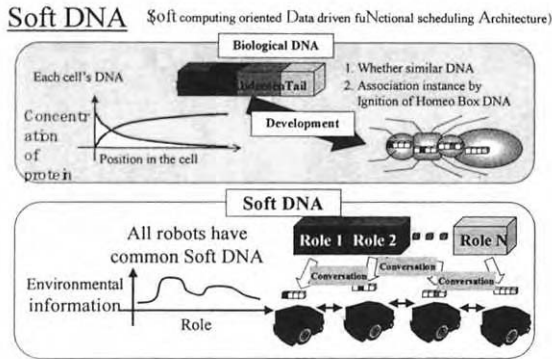


Figure 2: Soft DNA

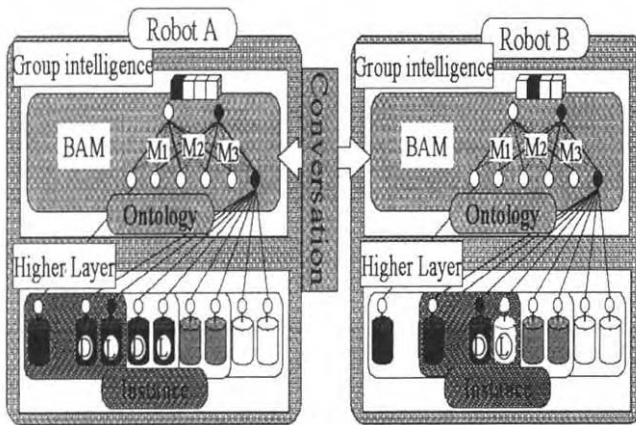


Figure 3

4. Robots intelligence structure and Control system

A robot recognizes a human face and hand by the camera located on the robot. The robot measures the distance toward the human from the face position on the display, and a human sign from the movement of the hand. Robots have a conversation with one another. The robot is controlled by an intelligent structure as shown in Fig.4, an intelligent structure based on Soft DNA and an intelligent model.

The robot practices a function according to the environmental information by group intelligence of the intelligent structure. The robot's role is divided into two types, a head and a chest. When a human orders a robot to come, the nearest robot becomes the head and follows the order. The others become the chest and stay. A higher layer treats practical control knowledge, associated by group intelligence. When the robot becomes a master, the information from the camera makes robots approach a human. At this point, the robot turns to make a human face near the center part of the display, i.e. if a human face is in the right side of the display, the robot turns right. When human faces y-coordinate on top of the display, robot estimates too close to the human and stop.

5. Experiment and result

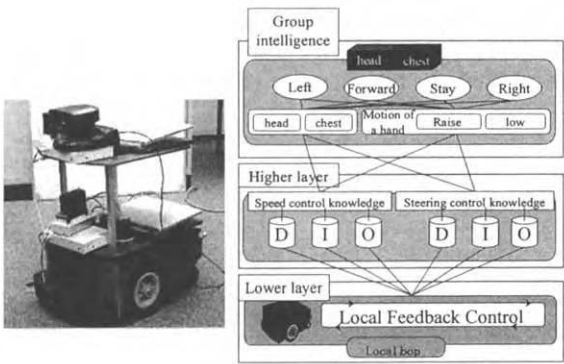


Figure 4: Robot's Intelligence

5.1 The entirety of system

In this paper, the robot uses a mobile robot 'lobo3', made in AAI, which puts on a note PC, a camera, and a wireless modem. The note PC takes a change in robot intelligence, the camera gets the state of a human and a wireless modem is used for a conversation between robots.

5.2 Experiment and result

In this paper, we experiment three situations. For all situations we use two robots, Rob1 and Rob2.

< Situation – 1 >

Both robots can watch a human face and a hand. The nearer robot will follow a human order, and approach to the human being. (Fig.5, Chart.1)

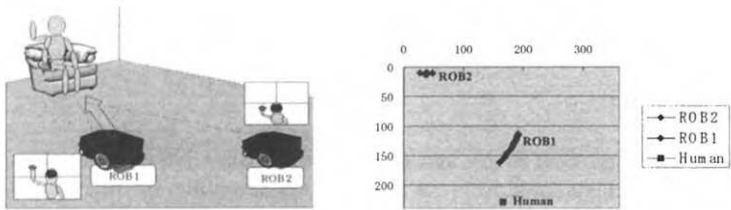


Figure 5

Chart 1

< Situation – 2 >

Rob2 is nearer to the human than Rob1. But there is a block in front of Rob2. Therefore Rob2 can't see the human, so can't understand the distance toward the human and the order. Rob2 does not know how to move. The other side Rob1 can see a human face and a hand. Rob1 becomes the head and follows the human order. (Fig.6, Chart.2)

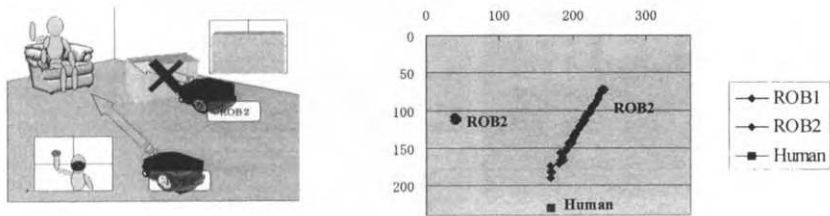


Figure 6

Chart 2

< Situation – 3 >

Rob1 is nearer to the human than Rob2. Rob2 can see a human face and a hand. Rob1 can see the human face, but can't see the hand. Therefore Rob1 can find out the distance to the human, but can't find out the human order. Rob1 and Rob2 share information of the human order. So Rob1 receives the order of common information in robots. Rob1 approaches the human based on the rule that nearer robots becomes the head. (Fig.7, Chart.3)

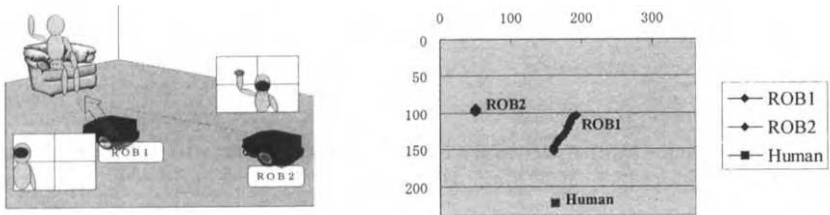


Figure 7

Chart 3

6. Conclusion

In this paper, we used Soft DNA and Ontological Neural Network for human centered intelligent robots, so they could follow a human order smoothly. Soft DNA and Ontological Neural Network are useful for a human support. From now on, we aim at a flexible reaction in many situations.

This research is supported by PRESTO, JST

References

- [1] T.Takagi et al.: Data retrieval Using Conceptual Fuzzy Sets, Proc. of The 9th IEEE International Conference on Fuzzy Systems. (FUZZ-IEEE2000), Vol.1 pp.94-99 (2000.5)
- [2] T. Yamaguchi, T. Yoshida, S. Mizuno, H. Hashimoto: Cooperative Works for Welfare Support Agents using Soft DNA, Proc. of The 8th IEEE International Conference on Fuzzy Systems (FUZZ-IEEE'99), Vol.3, pp.1454-1459(1999.8)
- [3] K. Ono, J. Miyamichi, T. Yamaguchi: Intelligent Robot System Using "Model of Knowledge, Emotion and Intention" and "Information Sharing Architecture", Proc. of 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA-2001), pp.498-501 (2001.7.29-8.1)
- [4] T. Yamaguchi, T. Kumazawa, H. Nitta: Cooperative Agent System Using Ontological Neural Network, Proc. of 2001 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA-2001), pp.508-510 (2001.7.29-8.1)

Trajectory Bounds of Solutions of Delayed Cellular Neural Networks Differential Systems

Ján Buša, Viktor Pirc

Faculty of Electrical Engineering and Informatics, Department of Mathematics,
Technical University, B. Němcovej 32, 041 20 Košice, Slovak Republic
{Jan.Busa, Viktor.Pirc}@tuke.sk
<http://www.tuke.sk/fei-km/index.htm>

Abstract. The aim of this paper is to present some mathematical methods for studying a specific class of cellular neural networks (DCNN) with delay described by a systems of differential equations with delays. We present some theoretical results which generalize the results of [2, 7, 8] for differential equations with delay. Some new criteria can be used to design networks and thus important significance in application. The results, similar to those in [1], are obtained by different methods.

Keywords: Cellular neural networks, Lower and upper approximation, De- lays

1 Introduction

We consider the following DCNN model described by system of nonlinear differential equations with delays

$$x'_i(t) = a_{ii}x_i(t) + \sum_{j=1}^n b_{ij}g_j[x_j(t)] + \sum_{j=1}^n c_{ij}g_j[x_j(t - \tau_j)] + k_i, \quad (1)$$

$$i = 1, 2, \dots, n,$$

where all a_{ii} are negative constants, b_{ij} , c_{ij} are nonnegative constants and k_i are real constants, $x_i(t)$ corresponds to the state vector of the i th unit at time t , $g_j[x_j(t)]$ denotes the output of the j th unit at time t , b_{ij} denotes the strength of the j th unit on the i th unit at time t , c_{ij} denotes the strength of the j th unit on the i th unit at time $t - \tau_j$, τ_j corresponds to the transmission delay along the axon of the j th unit and is nonnegative constant, a_{ii} represents the rate with which the i th unit will reset its potential to the resting state in isolation when disconnected from the network and external inputs $i, j = 1, \dots, n, t > t_0$ [1].

In [3, 4] the author investigates the case of time-varying coefficients a_{ii} and b_{ij} ($c_{ij} = 0$).

In [2] the authors investigate the case $g_j = g$ with the function $g : \mathbf{R} \rightarrow \mathbf{R}$

$$g(u) = -\frac{1}{1 + e^{-Gu}}, \quad (2)$$

where G is a positive constant. It is clear that $\forall u \in \mathbb{R}$: $-1 < g(u) < 0$, the function g is decreasing and $\max_{u \in \mathbb{R}} |g'(u)| = G/4$.

The initial value problem (IVP) for (1) is defined as follows: Let continuous initial vector function $\omega(t) = [\omega_1(t), \omega_2(t), \dots, \omega_n(t)]$ be given on the initial interval $\langle t_0 - \tau, t_0 \rangle$, where $\tau = \max(\tau_1, \dots, \tau_n)$. We have to find solution $\mathbf{x}(t) = [x_1(t), x_2(t), \dots, x_n(t)]$ of the system (1) defined on interval $\langle t_0 - \tau, \infty \rangle$ that satisfies the conditions

$$x_i(t) = \omega_i(t) \quad \text{for } t_0 - \tau \leq t \leq t_0, \quad i = 1, 2, \dots, n.$$

Definition: A constant solution $\tilde{\mathbf{x}} = (\tilde{x}_1, \dots, \tilde{x}_n)$ of algebraic system of equations

$$a_{ii}x_i + \sum_{j=1}^n (b_{ij} + c_{ij})g_j[x_j] + k_i = 0, \quad i = 1, 2, \dots, n, \quad (3)$$

is said to be an **equilibrium point for system (1)**.

Remark. Obviously, if for the solution of (1) it holds $\mathbf{x}(t) \equiv \tilde{\mathbf{x}}$ for all t in some interval of the length τ , then $\mathbf{x}(t) \equiv \tilde{\mathbf{x}}$ for all larger values t .

2 Preliminary results

In the following, we assume that each of the functions g_j ($j = 1, 2, \dots, n$) possesses the following properties:

- (H_1) g_j is bounded on \mathbb{R} ;
- (H_2) g_j satisfies Lipschitz condition (there is a number $\mu_j > 0$ such that $|g_j(u) - g_j(v)| \leq \mu_j |u - v|$ for any $u, v \in \mathbb{R}$);
- (H_3) g_j is unincreasing on \mathbb{R} .
- (H_4) there exists $q \in \mathbb{R}$ such that

$$\max_i \frac{M'}{|a_{ii}|} \sum_{j=1}^n (b_{ij} + c_{ij}) \leq q < 1, \quad (4)$$

where $M' = \max_j \mu_j$.

Remark. If the relations between the output of the cell and the state of the cell are described by functions $g_j = g$ given by (2) or by a piecewise-linear function $g(u) = -(1/2)(|u + 1| - |u - 1|)$, it is easy to see that each of the functions g_j satisfies the hypotheses (H_1)–(H_3).

Some results on the DCNN stability have been obtained in [1] using the Lyapunov functional method under the properties (H_1), (H_2):

Theorem 1 (Cao and Zhou). *For the DCNN (1), suppose that the output of the cell f_i , ($i = 1, 2, \dots, n$) satisfies hypotheses (H_1) and (H_2) above. Assume, furthermore, that the system parameters b_{ij} , c_{ij} ; ($i, j = 1, 2, \dots, n$) satisfy one of the following conditions:*

$$I. \quad \frac{\mu_i}{|a_{ii}|} \sum_{j=1}^n [|b_{ji}| + |c_{ji}|] < 1;$$

2. $\frac{1}{|a_{ii}|} \sum_{j=1}^n [\mu_j(|b_{ij}| + |c_{ij}|) + \mu_i(|b_{ij}| + \mu_i|c_{ji}|)] < 2;$
3. $\frac{1}{|a_{ii}|} \sum_{j=1}^n [\mu_j^2|b_{ij}| + |b_{ji}| + \mu_j|c_{ij}| + \mu_i|c_{ji}|] < 2;$
4. $\frac{1}{|a_{ii}|} \sum_{j=1}^n [|b_{ji}| + \mu_i^2|b_{ji}| + \mu_j|c_{ij}| + \mu_i|c_{ji}|] < 2;$
5. $\frac{1}{|a_{ii}|} \sum_{j=1}^n [\mu_j|b_{ij}| + \mu_i|b_{ji}| + \mu_j^2|c_{ij}| + |c_{ji}|] < 2$

where μ_j , $j = 1, \dots, n$ are constant numbers in the hypotheses (H_2) above.

Then the equilibrium \mathbf{x}^* of the DCNN (1) is globally asymptotically stable independent of delays.

3 Main results

Multiplying both sides of (1) by $e^{-a_{ii}t}$ and integrating from t_0 to t , we obtain

$$x_i(t) = x_i(t_0)e^{a_{ii}(t-t_0)} + \int_{t_0}^t e^{a_{ii}(t-s)} \left\{ k_i + \sum_{j=1}^n [b_{ij}g_j[x_j(s)] + c_{ij}g_j[x_j(s-\tau_j)]] \right\} ds, \quad (5)$$

where $i = 1, 2, \dots, n$.

Denote $M = \max_j \sup_u g_j(u)$ and $m = \min_j \inf_u g_j(u)$.

By definition, put for $t > t_0$

$$\phi_i^{(1)}(t) = x_i(t_0)e^{a_{ii}(t-t_0)} - \left[k_i + m \sum_{j=1}^n (b_{ij} + c_{ij}) \right] (1 - e^{a_{ii}(t-t_0)}) / a_{ii}, \quad (6)$$

$$\psi_i^{(1)}(t) = x_i(t_0)e^{a_{ii}(t-t_0)} - \left[k_i + M \sum_{j=1}^n (b_{ij} + c_{ij}) \right] (1 - e^{a_{ii}(t-t_0)}) / a_{ii} \quad (7)$$

and for $t \in \langle t_0 - \tau, t_0 \rangle$

$$\phi_i^{(1)}(t) = \psi_i^{(1)}(t) = \omega_i(t), \quad (8)$$

for $i = 1, 2, \dots, n$.

Lemma 1. $\forall t \in \langle t_0 - \tau, \infty \rangle$, $\forall i = 1, 2, \dots, n$:

$$\phi_i^{(1)}(t) \leq x_i(t) \leq \psi_i^{(1)}(t). \quad (9)$$

Proof. Using the lower and upper bounds of the functions $g_j(u)$ and the properties of the constants a_{ii} , b_{ij} and c_{ij} we get (9) from (5).

Theorem 2. Suppose each g_j satisfies the hypotheses (H_1) , (H_2) and (H_4) . Then there exists unique equilibrium point $\tilde{\mathbf{x}} = (\tilde{x}_1, \dots, \tilde{x}_n)$ of (1) and

$$\tilde{x}_i = \lim_{l \rightarrow \infty} x_i^{(l)}, \quad i = 1, 2, \dots, n,$$

where $\mathbf{x}^{(l)}$ is defined by recurrent process

$$x_i^{(l+1)} = -\frac{1}{a_{ii}} \left[k_i + \sum_{j=1}^n (b_{ij} + c_{ij}) g_j[x_j^{(l)}] \right], \quad l = 0, 1, \dots \quad (10)$$

with arbitrary starting vector \mathbf{x}^0 .

Proof. From (10) it is evident that

$$-\frac{k_i + m \sum_{j=1}^n (b_{ij} + c_{ij})}{a_{ii}} \leq x_i^{(l+1)} \leq -\frac{k_i + M \sum_{j=1}^n (b_{ij} + c_{ij})}{a_{ii}}, \quad i = 1, \dots, n.$$

Denote

$$D = \left\{ \mathbf{x} \in \mathbb{R}^n : \right. \\ \left. -\left(k_i + m \sum_{j=1}^n (b_{ij} + c_{ij})\right)/a_{ii} \leq x_i \leq -\left(k_i + M \sum_{j=1}^n (b_{ij} + c_{ij})\right)/a_{ii}, \quad i = 1, \dots, n \right\}$$

and let us consider next map $\mathcal{F}(\mathbf{x}) = [F_1(\mathbf{x}), \dots, F_n(\mathbf{x})]$:

$$F_i(\mathbf{x}) = -\frac{1}{a_{ii}} \left[k_i + \sum_{j=1}^n (b_{ij} + c_{ij}) g_j[x_j] \right]. \quad (11)$$

If $\mathbf{y} = (y_1, \dots, y_n)$ and $\|\mathbf{x} - \mathbf{y}\| = \max_j |x_j - y_j|$, by (10) and (H_2) we have

$$\begin{aligned} |F_i(\mathbf{x}) - F_i(\mathbf{y})| &\leq -\frac{1}{a_{ii}} \sum_{j=1}^n (b_{ij} + c_{ij}) |g_j(x_j) - g_j(y_j)| \leq \\ &\leq -\frac{1}{a_{ii}} \sum_{j=1}^n (b_{ij} + c_{ij}) \mu_j |x_j - y_j| \leq -\frac{M' \max_j |x_j - y_j|}{a_{ii}} \sum_{j=1}^n (b_{ij} + c_{ij}). \end{aligned}$$

Then we have

$$\|\mathbf{F}(\mathbf{x}) - \mathbf{F}(\mathbf{y})\| \leq \max_i \left[-\frac{M' \max_j |x_j - y_j|}{a_{ii}} \sum_{j=1}^n (b_{ij} + c_{ij}) \right] \leq q \|\mathbf{x} - \mathbf{y}\|.$$

It is clear that the map $\mathbf{y} = \mathcal{F}(\mathbf{x})$ is contractive on closed domain D , satisfies the conditions of Banach's theorem and has unique fixed point $\tilde{\mathbf{x}}$ on D (and on \mathbb{R}^n , too). The theorem is proved.

Remark. In [1] existence is obtained under conditions

$$\mathcal{F}(\mathbb{R}^n) \subset \mathbf{Q} = \{(x_1, \dots, x_n) \in \mathbb{R}^n : |x_i| \leq K, \quad i = 1, \dots, n\}$$

$$K = \max_i \left(\sum_{j=1}^n \left| \frac{1}{a_{ii}} (b_{ij} + c_{ij}) \right| M' + \left| \frac{k_i}{a_{ii}} \right| \right).$$

The uniqueness of the equilibrium follows from the global asymptotic stability proven in Theorem 1 under condition 1.

Let us consider the following map $\mathcal{G}[\mathbf{f}(t)] = [G_1[\mathbf{f}(t)], \dots, G_n[\mathbf{f}(t)]]$:

$$\begin{aligned} G_i[\mathbf{f}](t) &= f_i(t_0)e^{a_{ii}(t-t_0)} + \\ &+ \int_{t_0}^t e^{a_{ii}(t-s)} \left\{ k_i + \sum_{j=1}^n [b_{ij}g_j[f_j(s)] + c_{ij}g_j[f_j(s - \tau_j)] \right\} ds, \end{aligned} \quad (12)$$

for $t > t_0$ and for $t \in \langle t_0 - \tau, t_0 \rangle$ we define

$$G_i[\mathbf{f}](t) = \omega_i(t), \quad (13)$$

where $\mathbf{f}(t) = [f_1(t), \dots, f_n(t)]$, $t \in \langle t_0 - \tau, \infty \rangle$, $f_i(t) = \omega_i(t)$ for $t \in \langle t_0 - \tau, t_0 \rangle$, $i = 1, 2, \dots, n$.

Remark. Notice that according to (5) the solution $\mathbf{x}(t)$ of IVP problem for (1) is the fixed point of the map \mathcal{G} .

Further we will use the following notation:

$$\mathbf{f} \leq \mathbf{h} \quad \stackrel{\text{def}}{\iff} \quad \forall t \in \langle t_0 - \tau, \infty \rangle \forall i = 1, 2, \dots, n : f_i(t) \leq h_i(t).$$

Lemma 2. Let each function $g_j(t)$ ($j = 1, 2, \dots, n$) possess (H_3) . Then the map \mathcal{G} , defined by (12–13) is unincreasing, i. e.

$$\mathbf{f} \leq \mathbf{h} \quad \implies \quad \mathcal{G}(\mathbf{h}) \leq \mathcal{G}(\mathbf{f}).$$

Proof. This lemma can be proved by direct calculations, using the functions' $g_j(u)$ and the coefficients' a_{ii} , b_{ij} , c_{ij} properties.

Further, by definition, put for $t > t_0$, $l = 1, 2, \dots$,

$$\phi_i^{(l+1)}(t) = x_i(t_0)e^{a_{ii}(t-t_0)} + \quad (14)$$

$$+ \int_{t_0}^t e^{a_{ii}(t-s)} \left\{ k_i + \sum_{j=1}^n [b_{ij}g_j[\psi_j^{(l)}(s)] + c_{ij}g_j[\psi_j^{(l)}(s - \tau_j)] \right\} ds,$$

$$\psi_i^{(l+1)}(t) = x_i(t_0)e^{a_{ii}(t-t_0)} + \quad (15)$$

$$+ \int_{t_0}^t e^{a_{ii}(t-s)} \left\{ k_i + \sum_{j=1}^n [b_{ij}g_j[\phi_j^{(l)}(s)] + c_{ij}g_j[\phi_j^{(l)}(s - \tau_j)] \right\} ds,$$

where $\phi_j^{(l+1)}(t) = \psi_j^{(l+1)}(t) = \omega_i(t)$ for $t \in \langle t_0 - \tau, t_0 \rangle$.

Remark. $\phi^{(l+1)} = \mathcal{G}[\psi^{(l)}]$ and $\psi^{(l+1)} = \mathcal{G}[\phi^{(l)}]$.

Theorem 3. If x is the solution of (1), $\phi^{(l)}$ and $\psi^{(l)}$ are defined above, then for all $l = 1, 2, \dots$ it holds

$$\phi^{(l)} \leq x \leq \psi^{(l)}.$$

Proof. For $l = 1$ by Lemma 1 from (9) we have $\phi^{(1)} \leq x \leq \psi^{(1)}$. Therefore using Lemma 2 we have $\mathcal{G}[\psi^{(1)}] \leq \mathcal{G}[x] \leq \mathcal{G}[\phi^{(1)}]$. Hence $\phi^{(2)} \leq x \leq \psi^{(2)}$. In the same way we can prove the inequalities for all $l > 2$.

Theorem 4. Under the conditions of Theorem 3, for all $l = 1, 2, \dots$ we have

$$\phi^{(1)} \leq \phi^{(2)} \leq \dots \leq \phi^{(l)}(t) \leq \dots \leq x \leq \dots \leq \psi^{(l)} \leq \dots \leq \psi^{(2)} \leq \psi^{(1)}. \quad (16)$$

Proof. By (6) and (14), using the properties of each function $g_j(u)$ for $t > t_0$ we have

$$\begin{aligned} & \phi_i^{(2)}(t) - \phi_i^{(1)}(t) = \\ & = \int_{t_0}^t \sum_{j=1}^n \left[b_{ij} \left(g_j[\psi_j^{(1)}(s)] - m \right) + c_{ij} \left(g_j[\psi_j^{(1)}(s - \tau_j)] - m \right) \right] e^{a_{ii}(t-s)} ds \geq 0. \end{aligned}$$

Similarly, by (7) and (15), for $t > t_0$ we have $\psi_i^{(2)}(t) - \psi_i^{(1)}(t) \leq 0$.

Further, let for some k holds

$$\phi^{(k-1)} \leq \phi^{(k)} \leq \psi^{(k)} \leq \psi^{(k-1)}.$$

We have just showed, that it is true for $k = 2$. With respect to the properties of the functions $g_j(u)$ and constants a_{ii}, b_{ij}, c_{ij} we have for $t > t_0$

$$\begin{aligned} & \phi_i^{(k+1)}(t) = x_i(t_0)e^{a_{ii}(t-t_0)} + \\ & + \int_{t_0}^t \left\{ k_i + \sum_{j=1}^n \left[b_{ij} g_j[\psi_j^{(k)}(s)] + c_{ij} g_j[\psi_j^{(k)}(s - \tau_j)] \right] \right\} e^{a_{ii}(t-s)} ds \geq \\ & \geq \int_{t_0}^t \left\{ k_i + \sum_{j=1}^n \left[b_{ij} g_j[\psi_j^{(k-1)}(s)] + c_{ij} g_j[\psi_j^{(k-1)}(s - \tau_j)] \right] \right\} e^{a_{ii}(t-s)} ds + \\ & + x_i(t_0)e^{a_{ii}(t-t_0)} = \phi_i^{(k)}(t), \end{aligned}$$

where $i = 1, 2, \dots, n$.

Similarly as above, we get

$$\psi_i^{(k+1)}(t) \leq \psi_i^{(k)}(t), \quad i = 1, \dots, n.$$

Theorem 5. Suppose each g_j , $j = 1, 2, \dots, n$ satisfies (H_1) – (H_4) . Then for the sequences of continuous functions $[\phi_i^{(l)}]_{l=1}^\infty, [\psi_i^{(l)}]_{l=1}^\infty$ it holds

$$\lim_{l \rightarrow \infty} \psi_i^{(l)}(t) = \lim_{l \rightarrow \infty} \phi_i^{(l)}(t), \quad (17)$$

where $i = 1, \dots, n$ and $t \in \langle t_0 - \tau, \infty \rangle$.

Moreover, $\forall t \in \langle t_0 - \tau, \infty \rangle, \forall l \in \mathbb{N}$

$$\max_i |\psi_i^{(l)}(t) - \phi_i^{(l)}(t)| \leq \frac{M - m}{M'} q^l. \quad (18)$$

Proof. Because the sequences $[\phi_i^{(l)}(t)]_{l=1}^{\infty}$, $[\psi_i^{(l)}(t)]_{l=1}^{\infty}$ are bounded and monotonous, there exist $\lim_{l \rightarrow \infty} \phi_i^{(l)}(t)$ and $\lim_{l \rightarrow \infty} \psi_i^{(l)}(t)$ for all $t \geq t_0 - \tau$ and $i = 1, 2, \dots, n$.

Further, we prove (18) by induction on k . For $k = 1$ by (4), (6) and (7) we obtain for $t > t_0$

$$\max_i |\psi_i^{(1)}(t) - \phi_i^{(1)}(t)| \leq \max_i \frac{M - m}{|a_{ii}|} \sum_{j=1}^n (b_{ij} + c_{ij}) \leq \frac{M - m}{M'} q$$

and for $t > t_0$ we have also

$$\max_i |\psi_i^{(1)}(t - \tau) - \phi_i^{(1)}(t - \tau)| \leq \frac{M - m}{M'} q.$$

Now let us suppose that

$$\max_i |\psi_i^{(k)}(t) - \phi_i^{(k)}(t)| \leq \frac{M - m}{M'} q^k, \quad \forall t \in \langle t_0 - \tau, \infty \rangle.$$

Then $\max_i |\psi_i^{(k)}(t - \tau_i) - \phi_i^{(k)}(t - \tau_i)| \leq \frac{M - m}{M'} q^k$, $\forall t \in \langle t_0, \infty \rangle$. Thus for $t > t_0$ subtracting (14) from (15) we get

$$\begin{aligned} & \max_i |\psi_i^{(k+1)}(t) - \phi_i^{(k+1)}(t)| \leq \\ & \leq \max_i \int_{t_0}^t e^{a_{ii}(t-s)} \sum_{j=1}^n \left[b_{ij} \left| g_j[\phi_j^{(k)}(s)] - g_j[\psi_j^{(k)}(s)] \right| + \right. \\ & \quad \left. + c_{ij} \left| g_j[\phi_j^{(k)}(s - \tau_j)] - g_j[\psi_j^{(k)}(s - \tau_j)] \right| \right] ds \leq \\ & \leq \max_i \int_{t_0}^t e^{a_{ii}(t-s)} \sum_{j=1}^n \mu_j \left[b_{ij} \left| \phi_j^{(k)}(s) - \psi_j^{(k)}(s) \right| + \right. \\ & \quad \left. + c_{ij} \left| \phi_j^{(k)}(s - \tau_j) - \psi_j^{(k)}(s - \tau_j) \right| \right] ds \leq \\ & \leq \max_j \mu_j \frac{M - m}{M'} q^k \max_i \sum_{j=1}^n [b_{ij} + c_{ij}] \int_{t_0}^t e^{a_{ii}(t-s)} ds \leq \\ & \leq \frac{M - m}{M'} q^k \max_i \frac{M'}{|a_{ii}|} \sum_{j=1}^n [b_{ij} + c_{ij}] \leq \frac{M - m}{M'} q^{k+1} \end{aligned}$$

and

$$\lim_{k \rightarrow \infty} \max_i |\psi_i^{(k)}(t) - \phi_i^{(k)}(t)| \leq \frac{M - m}{M'} \lim_{k \rightarrow \infty} q^k = 0.$$

This completes the proof of Theorem 5.

References

- [1] Cao, J. — Zhou, D.: *Stability Analysis of Delayed Celluar Neural Networks*, Neural Networks 11 (1998) 1601–1605.
- [2] Buša, J. — Pirč, V.: *Trajectory Bounds of Solutions of Certain Systems of Nonlinear Differential Equations*, Procc. of Conf. 135 – Pannonian Applied Mathematical Meeting, Baia Mare-Borsa 2001 (*to appear*).
- [3] Daňo, I.: *Compatible Neural Networks*, Journal of Electrical Engineering, **50** (1999), No. 7–8, 203–205.
- [4] Daňo, I.: *Mathematical Notes on Neural Networks*, In: The State of the Art in Computational Intelligence (Eds.: Sinčák et al.), Proceedings of the European Symposium on Computational Intelligence, Košice, Slovak Republic, August 30 – September 1, 2000, 392–394.
- [5] Martinelli, G. — Perfetti, R.: *Neural Network Approach to Spectral Estimation of Harmonic Processes*, IEEE Proceedings – G, Vol. 140, No. 2, April 1993, 95–100.
- [6] Michaeli, L. — Šaliga, J. — Frič, T.: *Umelá neurónová sieť ako klasifikátor komplexného parametra*, Proceedings EDS 93, Brno, 139–141.
- [7] Michaeli, L. — Pirč, V. — Šaliga, J. — Frič, T.: *Rýchla metóda vyšetovania charakteristík dvojparametrického kvantizátora na báze analógových neurónových sietí*, Zborník konf. FEI TU, section Radioelectronics, Herlany (1994), 185–190.
- [8] Pirč, V.: *Some Properties of Systems of Nonlinear Differential Equations*, Bull. Appl. Math., Balaton (1994), 261–265.
- [9] Tank, D. W. — Hopfield, J. J.: *Simple "Neural" Optimization Networks: On A/D Convertor, Signal Decision Circuit and Linear Programming Circuit*, IEEE Trans. on Circuit and Systems, Vol. CAS-33, No. 5, May 1986, 533–541.

From Plain to Modular Topology: Automatic Modularization of Structured ANNs

Rudolf Jakša*

Kyushu Institute of Design

4-9-1 Shiobaru, Minami-ku, Fukuoka, 815-8540
JAPAN

jaksa@neuron.tuke.sk

Abstract. We propose automatic modularization method for artificial neural networks (ANNs). We treat modularization as an optimization task, therefore the optimization criteria are defined and the topology capable of continuous iterative modularization is introduced. The modularization process starts with unstructured plain network topology and iteratively builds up a modular structure. Automatic modularization approach not only learns to map inputs to outputs but it also tries to discover a structure of knowledge represented by training patterns.

1 Introduction

Structured, or modular, neural networks (ANNs) are common type of ANNs, however, the structure of these networks is usually fixed. Various methods exist for the optimization of structure (topology) for unstructured (plain) neural networks, these include pruning algorithms, ANNs evolved by GA (genetic algorithm), cascade correlation algorithm, etc. However, these algorithms do not produce a modular or hierarchical structure of network, they only reduce each-to-each connection scheme to more sparse network topology. They are no established methods/algorithms for modular ANNs with variable topology yet.


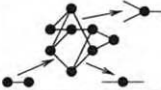
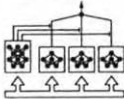
	fixed topology	variable topology
plain topology	 classical multilayer networks	 pruning, genetic alg., etc.
modular topology	 NARA, mixtures of experts	?

Figure 1: Types of the structure of ANNs.

*This work is supported by Japan Society for the Promotion of Science (JSPS).

Structured neural networks are usually of NARA[2] type or “Mixture of Experts”[3] type (see Fig. 3.). Both these types of ANN structures have similar topology. This topology consist of several parallel “expert” modules and a single antecedent module. Expert modules are specialized for particular subtasks of the whole task. An antecedent module controls a participation of expert modules on the final output of the network according to the position of an actual input in the input space. Such a modular structure is easy to follow for humans and allows partial understanding of the functionality of an ANN or embedding of an a priori knowledge into ANN. This type of structure is even used for a rough modeling of brain functionality, where several areas of cortex are responsible for several different tasks (they correspond to expert modules).

2 Automatic Modularization Approach

2.1 “Soft Module” Idea

NARA modular networks consist of several independent modules, where nodes in one module are not connected to nodes in other modules. In the plain ANN with each-to-each connection scheme every node depends on all other nodes (resp. on all nodes in the previous layer). In process of evolving modular network from plain network, it is logical to go through an interim state, where the nodes inside particular module are connected tightly and the nodes from two different modules have weak connections (see Fig. 2.).

Let define the dependency of two nodes to be corresponding to the strength of connection between them. This connection can be direct or indirect using intermediate nodes. In a simplified case we can consider two nodes with a small weight on their direct connection to be less dependent as those with a big weight on this connection. By this assumption, a module is a group of nodes with big weights on connections inside the module (intramodule connections) and with small weights on connections to other modules (intermodule connections). This idea of “soft module” is illustrated on Fig. 2. by a “structured plain” structure (dotted lines represent links with small weights).

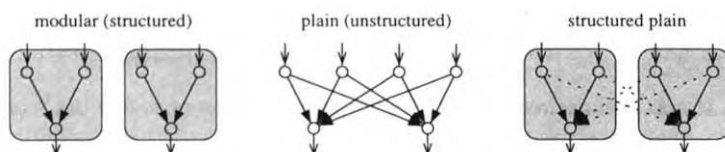


Figure 2: Types of ANN modularity; “structured plain” topology.

2.2 Passive Versus Active Approach to Modularization

Two basic approaches may be used to discover a modular structure of ANN: passive and active. These two approaches can be characterized:

Passive: It is applied after the learning of network. Analysis is done to find modular structure. Network is not modified.

Active: It is applied during the learning. Modularization algorithm is run and modular structure is intentionally iteratively developed. Network is modified.

Passive approach may be based on ANN analysis algorithms (for instance [6]) and is similar to the analysis of source code in reverse-engineering field [5]. Discovered ANN structure cannot be further optimized using a passive approach, while an active approach allows real structure optimization.

2.3 Optimization Criteria for Modularization

Both, passive and active, approaches need optimization criteria to be defined. In the passive approach this criterion is necessary for finding the best partitioning of a fixed network (see [5]). In the active approach this criterion is optimization criteria, used to “grow” modules inside of the network.

Simple optimization criteria for minimization of intermodule connections for single module m can be defined as:

$$J_e^m = \sum_{e=0}^{N_e^m} |w_e| \quad (1)$$

where N_e^m is the number of intermodule connections of module m and w_e are weights on these connections. Value J_e^m is zero if module m is disconnected from other modules.

Optimization criteria for maximization of intramodule connections for single module m can be defined as:

$$J_i^m = - \sum_{i=0}^{N_i^m} |w_i| \quad (2)$$

where N_i^m is the number of intramodule connections of module m and w_i are weights on these connections. Value J_i^m is zero if module m has no internal connections and it is less than zero if intramodule connections appear. Rule (2) does not distinguish between the following two situations:

- a) a single very big weight and a couple of small weights,
- b) a lot of moderately big weights.

Case b) represents what we actually want. To avoid case a) the criterion function (2) must be modified. We can simply ignore very big weights and do not include them into rule (2). This can be realized by limiting weights to fixed maximal value. Modified criterion function is:

$$J_i^m = - \sum_{i=0}^{N_i^m} |f_i(w_i)| \quad f_i(w) = \begin{cases} w & , \text{ if } |w| < w_{max} \\ w_{max} & , \text{ otherwise} \end{cases} \quad (3)$$

where w_{max} is maximal desired weight. Further maximization of such a big weight will not result in better evaluation now.

Even the criteria (3) is not ideal. It will not prevent possible emergency of intramodule structures not related to the solved task, instead only to comply with this criterion. For instance, some elimination structures may emerge, where big positive and negative weights are used only to eliminate themselves. Further work on analysis of ANN structures is necessary to avoid such risks as well as for understanding of structuralization in ANNs. Different approaches for building-up of strong intramodule connectivity can be used as well. For instance, pruning algorithms to eliminate weak connections inside modules or “Branch Control” approach [8] may be used to determine functionally related nodes and to group them into modules.

Criteria (1), (2) and (3) are based on described simplifications and on the assumption that dependency of modules is based only on direct connections between nodes. These criteria should be understood as basic criteria with possible further extensions.

2.4 Optimization Methods for Modularization

Described criteria may be minimized using a custom minimization algorithm or using a general optimization method which is able to deal with arbitrary optimization criteria. Although development of a simple custom algorithm dealing with criteria (1) and (3) may be easy, optimization should also deal with original task criteria (usually approximation or classification) and it should allow next extensions of these criteria. Algorithms able to optimize according to arbitrary criteria may be advantageous here. Several well-known approaches may be used for optimization according to arbitrary criteria: random search, evolutionary computation (EC) or adaptive critic (AC) method.

2.5 Weighting the Output of Modules

NARA type networks contain an antecedent module which controls participation of other modules on the final output. The antecedent module is the “IF part” module in NARA and the “gating” module in “Mixture of Experts” networks. Control of participation of other modules is provided by the weighting of outputs of these modules. Weighting in NARA is realized by application of (fuzzy) operators on outputs from modules. However, this weighting can be also provided by regular connections to output nodes of particular modules (see Fig. 3.). Sufficiently intensive inhibitory signal on such a connection can effectively stop participation of a given module on the final output and a zero signal will switch the module to the full participation on the output. This is equivalent for the 0 and the 1 weighting signals in NARA network.

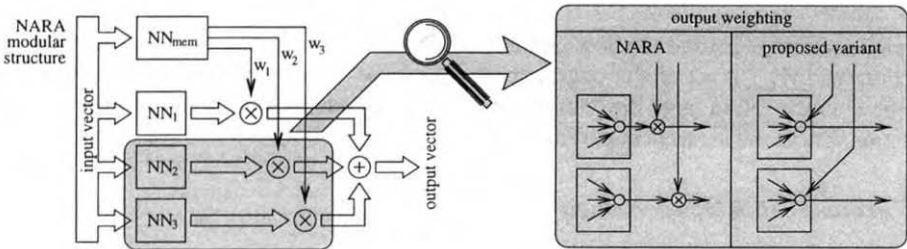


Figure 3: Weighting of outputs of modules in NARA and proposed type of weighting.

Outputs of nodes in ANNs with sigmoidal activation functions $f(in) = 1/(1 + e^{-in})$ are always positive. To obtain big a inhibitory signal in such a network it is necessary to use big negative weights on the particular connection. This means that antecedent network should use connections to other modules with big negative weights (see Fig. 4.).

2.6 Hierarchical Modular Structures

The described type of output weighting allows a construction of a hierarchical modular structure. This can be obtained by allowing all the modules to control participation of other modules on the final output (Fig. 4.) and it will allow iterative emergency of hierarchical modular structures (like in “Hierarchical Mixtures of Experts” presented in [4]). In the exam-

ple on Fig. 4. the module 1 is superior in hierarchy to modules 3 and 4, and module 3 is superior to module 2.

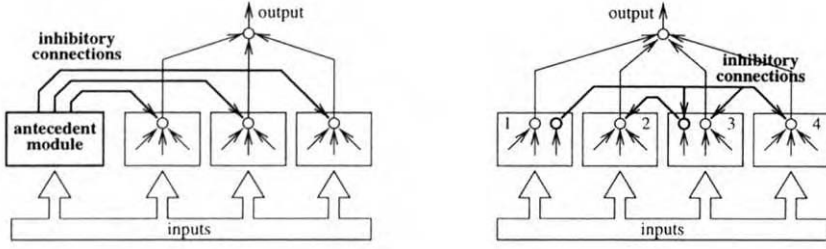


Figure 4: Structure based on the explicit antecedent module versus hierarchical modular structure without explicit antecedent module.

The Discussed type of a hierarchical modular structure can be characterized by:

- 1) uniform modules with no explicit antecedent module,
- 2) tight internal connections in every module,
- 3) weak intermodule connections,
- 3a) except few strong inhibitory intermodule connections.

These characteristics can be projected to the following change of criteria (1):

$$J_e^m = \sum_{e=0}^{N_e^m} |f_h(w_e)| \quad f_h(w) = \begin{cases} 0, & \text{if } w \in W_{inh} \\ w, & \text{otherwise} \end{cases} \quad (4)$$

where function f_h is used to allow a limited number of strong inhibitory intermodule connections and W_{inh} is set of n strongest negative intermodule connections. These connections are now an exception from the “punishment” of intermodule connections and they provide functionality of the antecedent part of NARA.

2.7 Parameters of Modularization

There are two important parameters of a modular network: number of modules and size of modules (number of nodes in every module). These parameters may be set fixed or may be adapted on-line. For the on-line adaptation of parameters, an algorithm [5] from the research of reverse-engineering of a source code can be used: start with random partitioning of network and iteratively try similar partitions in order to find one which minimizes structure optimization criteria better. This algorithm can be used for passive structure discovery in a fixed network, as well.

3 Experiments and Discussion

Experimental setup on Fig. 5. was used for the proof-of-concept of a proposed method. The Structure with an explicit antecedent module and fixed number of expected modules was defined. An adaptive critic method was used for optimization. The main task was classification of two-dimensional data into two classes.

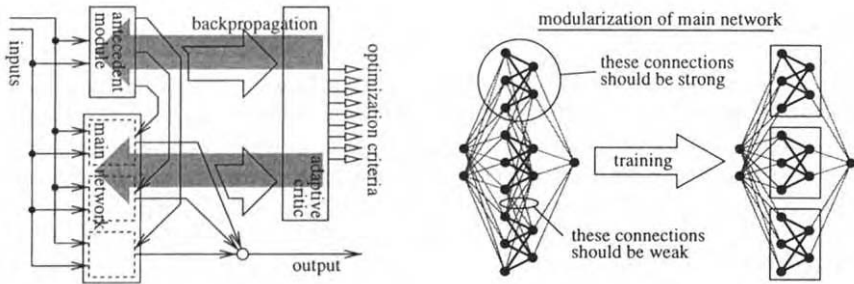


Figure 5: Experimental setup and modularization of main network.

The goal of the experiment was to change plain network (“main network” on Fig. 5.) to modular network. This can be indicated by observing of minimization of intermodule connections and the growth of intramodule connections during the learning. Optimization criteria for fixed number of modules were defined independently. There were defined two criteria for each expected module: one for minimization of intermodule connections and second for maximization of intramodule connections. Other two criteria were defined for maximization of inhibitory connections from the antecedent module and for minimization of a classification error. There were 10 optimization criteria altogether.

Scheme in the Fig. 5. is complicated, as it consists of several modules and several signal flows. However, this setup was chosen in order to test “modularization” ability of proposed method. Structure in Fig. 4. and adaptation of module parameters should allow further refining and simplification of the technique. Ideally, only two explicit modules should be used: unstructured main network and adaptive critic network.

The results of the experiment are positive in sense of observing modularization process. The experiments also showed high importance of usage of efficient multiobjective optimization technique for proposed type of iterative ANN modularization. In realistic situations the proposed method involves relatively high number of optimization criteria and concurrent optimization of all the criteria should be done. This multiobjective character favors the evolutionary computation compare to adaptive critic method for structure optimization in this task.

The time order in which criteria are satisfied can be important for overall results too. Development of crisp modules prior to development of task solving abilities of network will probably lead to different modularization than a reverse order. This “timing” as well as weighting of criteria is important to allow to focus the optimization on different aspects of modularization.

Automatic modularization methods can simplify application of modular ANNs in real-world tasks. The modularization can also change the “black-box” behavior of ANN to “grey-box” behavior, as the structured ANN can be better analyzed than traditional plain networks. Modularization can be understood as some kind of unsupervised learning running concurrently with the main learning of network. This unsupervised learning is based on idea of expectation of modular structure of knowledge and it results in such a modular structure. Modularization of structure is an alternative method of the ANN simplification when compared to elimination of links and nodes from ANN. In this sense, modularization is an alternative method for improvement of generalization performance of ANNs.

4 Conclusion

A method for automatic modularization of neural networks was proposed in this paper. The idea of “a soft module” in contrast to strictly independent modules was introduced. This idea allowed us to treat modularization as the optimization task, where optimization is the optimization of weights. The optimization criteria for this task of modularization were defined and selection of proper optimization method was discussed. Realized experiments emphasized a strong multiobjective character of the task and we recommend to focus on this aspect of method in future work. We assume automatic modularization of ANNs an important technique for study of generalization and analysis of function of ANNs as well as for further improvement of ANNs performance.

References

- [1] R. Jakša Automatic Modularization of ANNs Using Adaptive Critic Method, *Proceedings of the 3rd WSEAS Conference on Neural Networks and Applications (NNA'02)*, February 11-14, 2002.
- [2] H. Takagi, N. Suzuki, T. Koda, Y. Kojima, Neural Networks Designed on Approximate Reasoning Architecture and Their Applications, *IEEE Transactions on Neural Networks*, Vol.3, No.5, September 1992, pp. 752-760.
- [3] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, G. E. Hinton, Adaptive Mixtures of Local Experts, *Neural Computation*, No.3, 1991, pp. 79-87.
- [4] M. I. Jordan, R. A. Jacobs, Hierarchical mixtures of experts and the EM algorithm, *Neural Computation*, No.6, 1994, pp. 181-214.
- [5] S. Mancoridis, B. S. Mitchell, C. Rorres, Y. Chen, E. R. Gansner, Using Automatic Clustering to Produce High-Level System Organizations of Source Code, *IEEE Proceedings of the 1998 Int. Workshop on Program Understanding (IWPC'98)*, 1998.
- [6] R. Aharonov-Barki, I. Meilijson, E. Ruppín, Who Does What? A Novel Algorithm to Determine Function Localization, *NIPS'2000*, Vol.13, 2000.
- [7] R. S. Sutton and A. G. Barto, *Reinforcement Learning*, MIT Press, 1998.
- [8] Q. Xiong, K. Hirasawa, J. Hu, J. Murata, A study of Brain-like Neural Network Model, *Knowledge-Based Intelligent Information Engineering Systems & Allied Technologies (KES'2001)*, 2001, pp. 303-307.

Neural Structure as a Modular Developmental System

Eva VOLNÁ

University of Ostrava, 30 dubna st. 22, 701 03 Ostrava, Czech republic
eva.volna@osu.cz

Abstract. This paper is focused on a long-term goal of developing neural networks with a modular architecture. The contribution is aimed at a design of a general system that would adapt its behavior in an environment. The consistent behavior would have to emerge from an interaction of the system with its environment, which should be achieved by an adaptation of a system by evolutionary algorithms, so that modular neural structures would be constructed and a behavior would emerge similarly as in natural evolution.

Keywords: Emergence, Evolutionary Algorithms, Neural Networks, Modularity.

1. Evolutionary algorithms and artificial neural networks

Conventional evolutionary algorithms [1] are optimization algorithms inspired by natural evolution. They perform well at a wide range of difficult optimization problems, which is currently what most evolutionary algorithms are used for.

Evolutionary algorithms evolve an initially randomly generated population of solutions by selecting individuals (solutions), which will pass into the next generation. To do this they evaluate the individuals' fitness via some procedures relevant to the problem. The fitter individuals reproduce, replacing less-fit individuals, which perish. Then the cycle starts over again, starting with the resulting population from the last generation. In most evolutionary algorithms, the population size remains constant. There is a mutation operator used at reproduction in this paper. There is not a crossover operator used in the described algorithm, because the convergence is then better (e.g. this algorithm looks like hillclimbing algorithm).

Conventional artificial neural networks [2, 3] are an attempt to produce systems that work in a similar way to biological nervous systems. Conventional artificial neural networks contain a number of simple processing units. Each unit has an output, which may be propagated to other units via a weighted link, and a number of inputs from other units. The inputs to a unit are each multiplied by the weight of the relevant link and then summed to produce the total input to the unit. The unit's output is then a function of this total input and unit constants, such as threshold. The structure of most artificial neural networks has traditionally been layered, with all of a unit's links directed to units in higher layers. We can talk of an input layer and an output layer, with all intermediate layers being hidden layers. Learning in these artificial neural networks is most commonly achieved via supervised backpropagation. The multi-layer structure of artificial neural networks is used in this paper.

2. Emergence and modularity in artificial neural networks:

Emergence is relevant to modular neural nets. Emergence relates to unexpected, unprogrammed behaviors. However, this is not the best way how to define emergence, because it depends on the predictive ability of the observer and demands a once-only instance of emergence. Emergence is used in the paper [4] to refer to ongoing processes, which produce results requiring vocabulary not previously involved in the description of the system's inner components. This is the meaning used throughout this paper.

Aiming at such an emergence by a utilization of evolutionary algorithms seems to be the most promising route to building a device, which outperforms the specifications that we give it. This should coincide with an emergence and development of modular neural network architecture.

The modular architecture consists of combination of several conventional networks. There are currently three main approaches to the modular development of artificial neural networks [5]: cellular encoding, cellular automata and Lindenmayer systems.

3. Evolving modular neural net architecture via evolutionary algorithms

A method is described in which the network evolves a modular architecture, where each module represents a comparably independent part of the network. The network at the same time learns the training patterns. An outcome of this learning should be that different parts of the network learn different training patterns and, thus, learn to compute different functions. The architecture should perform task decomposition in the sense that it learns to partition a task into two or more functionally independent tasks allocated to distinct network parts (modules) where each learns its task. The method is tested on real world examples.

The method used for a construction of such a network is based on the evolutionary algorithms. The same evolutionary algorithms map a given problem onto a set of strings, each string representing a potential solution. A string encodes the modular network architecture. The selection of a suitable neural network topology for a solution of the task is a problem. The neural network topology must correspond to the complexity of task, e.g. the number of training patterns, its inputs and outputs and the structure of relations that are described. Each network architecture should be encoded in the genotype representation and the weight values of these networks should be determined by backpropagation. The modular network architecture for a given task then emerges as a winning architecture obtained by evolutionary algorithms. Each chromosome (e.g. the network architecture) is in this algorithms adapted and evaluated on the basis of teaching. Three kinds of information are included in fitness function: parameters of network architecture, its net error (we want to minimize the sum of square of deviation of output from neural network) and speed of its teaching that is used in a final condition for a calculation's terminating. Numbers of units in the input and output layers and their ordering are determined by the data from a training set and the number and the ordering of connections among units in the hidden and the output layer is the problem that is solved in this paper. Hidden units can fall into one or more modules of network architecture.

If the task is decomposable, it is not suitable for units to fall into more modules. We have defined such conditions in a fitness function and in the learning procedure, that the undesirable interference between modules is rejected during the development of the architecture. We do not build modules in neural net emerge straightforwardly by penalizing interference between modules. The description of this method forming the modular architecture of neural nets is the main topic of this paper. The mentioned method will be

demonstrated for a real task solving (e.g. The What and Where Vision Tasks described in [6]).

The following part describes a method of optimization of the modular neural network architecture via evolutionary algorithms: First, we must propose the architecture of neural network (e.g. number of input, hidden and output units) before the main calculation.

Thereafter the process of evolutionary algorithms is applied. With its assistance the best optimal population starts from the set of individuals that are generated randomly. The search for optimal population is ended when the population achieves the maximal generation or if the fitness function of a individual and at the same time its speed teaching¹ achieves the maximal values over ten successive generations.

The population consists of individuals. Every individual is described by his chromosome (see Fig. 1), where m is number of hidden units n is number of output units. There is $e_{ij} = 0$, if a connection does not exist and $e_{ij} = 1$, if a connection exists ($i = 1, \dots, m, j = 1, \dots, n$) in the every chromosome. Connections between input and hidden units are not included in the chromosomes, because they are not necessary for a creation of modular network architecture.

POPULATION:

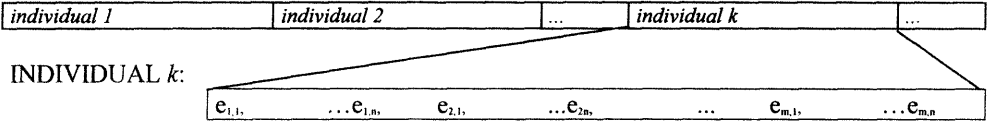


Figure 1. Population of individuals.

For every individual, an error (E_i) between the desired and the real output in a procedure which implements the forward distribution of signals in a multi-layer neural network is calculated. On the basis of this, the fitness function is calculated as follows:

$$Fitness_i = \begin{cases} k \cdot \frac{1}{E_i} & \text{if } connection_i > max_connection, \\ \frac{1}{E_i} & \text{otherwise.} \end{cases}$$

where $i = 1, \dots, p$ (p = number of individuals in the population);

$connection_i$ is a real number of connections among units in the hidden and the output layer;

$max_connection$ is a constant equal to maximal number of connections among units in the hidden and the output layer;

k is a defined constant ($0 < k < 1$).

The mutation operator is defined in the following way: For every individual each connection between hidden and output layers is changed with probability 0,01 (e.g. if the connection exists – after mutation it does not exist and vice versa). For every individual after mutation its fitness value is calculated and only better individuals or its mutation is

¹ The individual is adapted by backpropagation and the speed of teaching is calculated as a difference of net error in the i -th and j -th step ($0 < i < j < max_step$, where max_step is a maximal defined number of teaching steps).

included into the next generation. Next, one individual (in every generation) is randomly chosen and mutation operator is applied.

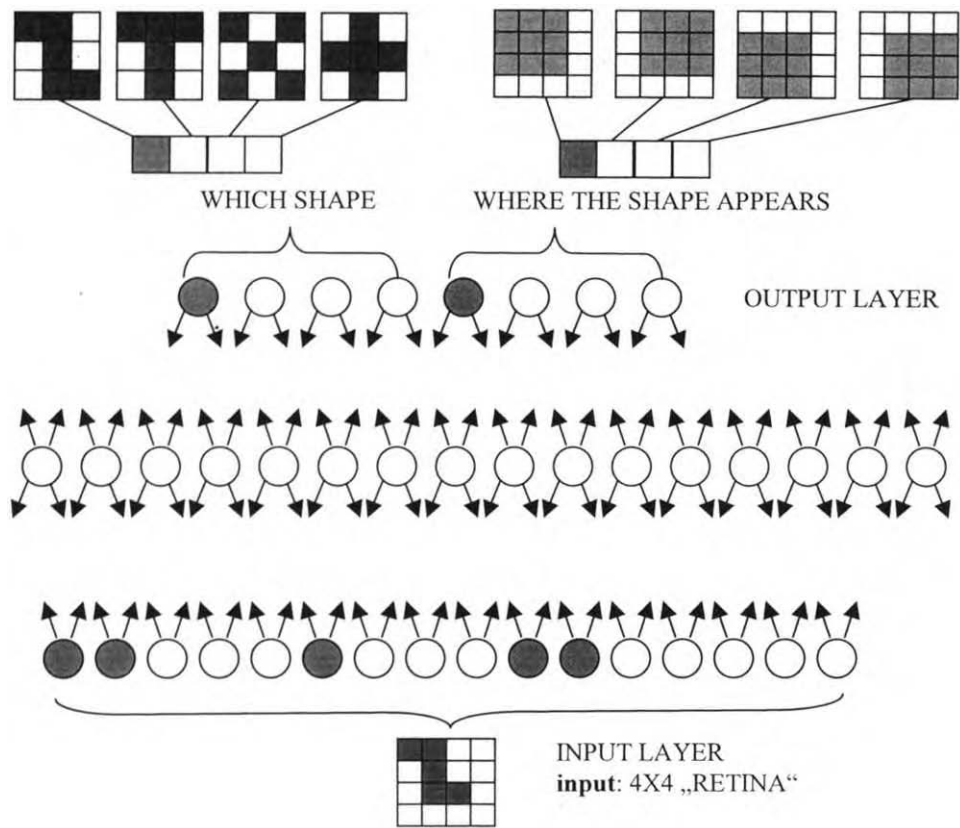


Figure 2. Structure of the model

4. Structure of the models

The present paper is an investigation of the computational demands on a single system that represents both shape and location versus two systems with two kinds of information being represented separately [6].

The model is a feed-forward network with 16 input nodes, 16 hidden nodes, and 8 output nodes. As illustrated in Figure 2, the 16 input nodes are organized into 4x4 matrix, and patterns are represented by selectively activating specific input nodes. Each hidden node receives an input from all 16 input nodes, and each hidden node in turn sends an input to all 8-output nodes. Four shapes are formed by activating different combinations of input nodes, as illustrated in Figure 2. These shapes are defined as different combinations of cells within 3x3 grid. The 3x3 grid for each shape is centred at any of four different locations on the 4x4 “retinal” input grid. Thus, there were 16 different combinations of shape and location, and each combination is represented by a binary pattern so that a node is turned on if its corresponding cell in the input array is covered by the shape and turned off otherwise. The 8 output nodes are divided into two subsets of four nodes each. Nodes in the “what” subset are responsible for indicating of the identity of the input. Each shape is associated with one of the four “what” nodes, and each of the four locations at the which a shape could

be presented is associated with one of the four “where” nodes, and the system is considered to correctly locate an input.

5. Conclusion

The unsplit model, a one-system mechanism, is a fully interconnected feed-forward network with input, hidden and output units; e.g. each hidden node is connected to all of the output nodes, and each output node receives a connection from each of the hidden nodes. The associated split model has the same number of units in the input, hidden and output layers. The input layer is fully interconnected with the hidden layer. The structure of the split model differs from a structure of the unsplit model only in the pattern of connectivity between the hidden and output layers. In the split model is the pattern of connectivity between the hidden and output layers restricted by partitioning the hidden nodes into two subsets. Nodes in one of the subsets are connected only to the “where” output nodes, and nodes in the other subset of the hidden layer are connected only to the “what” output nodes. Thus in the split model, each hidden node sends input to only half of the output nodes, and each output node receives input from only half of the hidden nodes. The split model thus functions as two distinct systems, overlapping only at the input level.

Above described method can rebuild the split model from the unsplit model. We started with the population of individuals that are generated randomly. An evolution of the best individual in the population is shown in Figure 3. There is only one modul in the first generation and six moduls in the last generation. Three units realise the “what” problem and three units realise the “where” problem. The next two units realise the “what” problem with probanility 0,875 and with probability 0,75 two units realise the “what “ problem and two units the “where” problem. The rest of units are not specified.

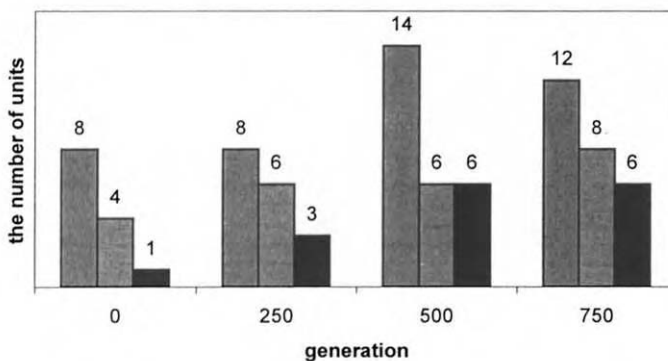


Figure 3. A graph of an evolution of the best individual in the population.

References

- [1] Goldberg, D. E.: Genetic algorithm in search optimalization and machine learning. Addison-Wesley, Reading., Massachusets 1989.
- [2] Beale, R. - Jackson, T.: Neural Computing: An Introduction. J W Arrowsmith Ltd, Bristol, Great Britain 1992.
- [3] Fausett, L. V.: Fundamentals of neural networks. Prentice-Hall, Inc., Englewood Cliffs, New Jersey 1994.
- [4] Cho, S. B. – Shimohara, K.: Emergence of structure and function in evolutionary modular neural networks. Proc. Fourth European Conf. on Artificial Life (ECAL'97), pp.197-204 (1997).

- [5] Kvasnička, V. – Pospíchal, J. – Tiňo, P.: Evolutionary algorithms. STU Press, Bratislava 2000; in Slovak.
- [6] Rueckl, J. G.: Why are “What” and “Where” processed by separate cortical visual systems? A computational investigation. *Journal of Cognitive Neuroscience* 2 (199?), 171-186.

PREDICTION SYSTEMS BASED ON FIR NEURAL NETWORK WITH CORRECTION OF ERROR

Daniel Novotný¹, Peter Sinčák^{1,2}

¹*CIT – Center for Intelligent Technologies,
Department of Cybernetics and Artificial Intelligence,
Faculty of Electrical Engineering and Informatics,
TU Košice, Letná 9, 04001 Košice, SLOVAK REPUBLIC
cig@neuron.tuke.sk, <http://www.ai-cit.sk>*

²*Siemens PSE& I, AG Vienna, ECANSE Group, Austria
Peter.Sincak@siemens.at*

Abstract. The paper deals with research and experience of application of FIR filter (Finite Impulse Response) in BP neural networks used for prediction system. Prediction can be defined as extrapolation of unknown function determined by representing data sets. The cascade approach is used to improve the prediction process. It is possible to receive error signal and use it to correction model. Comparison between FIR neural network and FIR neural network with correction models is described. The results show useful approach considering studied prediction of the predictor error with the aim to improve the overall performance of the prediction system.

Keywords: prediction, FIR, finite impulse response, correction of error, neural networks, function approximation,

1. Introduction

Intelligent technologies represent a very interesting direction in current and future technology. These technologies are based on AI techniques including neural networks and other intelligent approaches. Prediction systems are very important in many domains and have extremely wide application potential in several applications. Prediction problems are actual in any part of everyday life including technological processes in industrial as well as non-industrial domain. Since prediction as function extrapolation of the unknown function is a data depended problem, the development of the universal prediction system is very difficult. This project shows the approach of designing flexible and adaptive tools based on a prediction system, which consists of the predictor and prediction error correction module.

2. Project motivation and basic concept

Prediction is in fact an extrapolation of the unknown function, which is being approximated by neural network or any statistical tools. Usually we have in the input

$$(\mathbf{x}(t), \mathbf{x}(t-1), \dots, \mathbf{x}(t-\tau)) \quad (1)$$

where τ - determines the width of the so-called input window size and output $(\mathbf{y}(t+1), \mathbf{y}(t+2), \dots, \mathbf{y}(t+\lambda))$, where λ - is size of the length of the time series of the predicted variable and \mathbf{x} and \mathbf{y} are vectors. Always the $\tau \gg \lambda$ and the easiest situation is when $\lambda=1$ or only

[†] This project is supported by Vega Project from Ministry of Education of Slovak Republic "Intelligent Technologies in Modeling Intelligent Systems" and also partially by Maria-Curie Individual Fellowship.

one value is predicted. Certainly problems if $\lambda \gg 1$ are more difficult and prediction task is more complex and difficult.

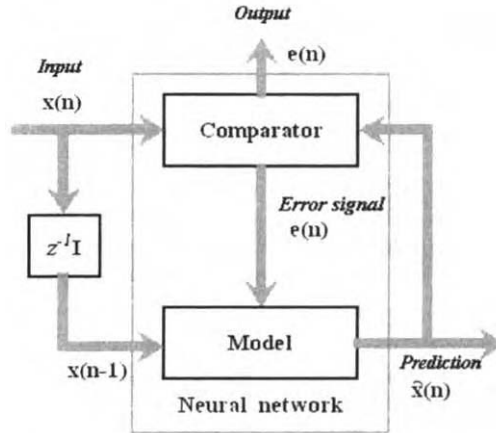


Figure 1 Prediction system

Figure 1 shows concept of the neural network predictor. Part *Model* estimates value $\hat{x}(n)$ in next step using values from the previous steps $x(n-1)$ and parameters of network calculated in time $n-1$. In most cases, predicted value and real value is different, then the part that is called *Comparator* calculates the difference between $x(n)$ and $\hat{x}(n)$. Computed error signal $e(n)$, also called *innovation* signal and presents new information in input signal $x(n)$. If $e(n)$ is zero, input signal does not contain new information and the model estimated output value correctly. If $e(n)$ is non-zero, It means that error signal contains new information and model should be modified.

Innovation signal that is obtained from *comparator* should be used:

1. It is a correction signal that adapts model (synaptic weight in neural network).
2. It is one of outputs of the prediction system that can be presented to next level for further processing.

3. FIR neural network predictor

Application of Error back propagation includes temporal error BP approach to the FIR neural training. FIR is Finite Impulse Response filter implemented in BP feed-forward neural network and has an important role in estimation of the needed historical data for prediction tasks [1]. If we assume that the normal BP neural networks were based on regular Error function as follows:

$$J^p = \frac{1}{2} \sum_{i=1}^{N_o} (ev_i^p - x_i^p)^2 \quad (2)$$

here ev_i^p is expected value on the neural network for i -th output neuron concerning input pattern "p". Also x_i^p represents the real output of on i -th output neuron concerning input

pattern “p” while N_0 is number of neuron in the output layer. Then a usual BP- like adaptation rule is used as follows:

$$\Delta \bar{w}_{ij}^l(n) = -\gamma \bar{\delta}_i^{l+1}(n) \bar{x}_j^l(n) \quad (3)$$

where

- I pre-synaptic neuron
- j post-synaptic neuron
- n iteration step
- w synaptic weight
- γ is learning parameter of learning
- δ_i^{l+1} is an error signal associated to neuron “j” which is in layer “l+1”
- x_i^l is value of activation of pre-synaptic neuron “i” in layer “l”

There is a basic difference between a regular BP approach and the one applied for FIR neural network; as follows:

- It is associated a vector synaptic weight to each synapse.
- The propagation of Error is being effective after number of inputs, which will propagate through the overall neural network up to the end. Only after then an Error will occur and can be propagated back to the neural networks.

Concerning the FIR filters on neural network which have length of the FIR filter “M” and there is L layer of the neural networks then we can express the adaptation as follows:

$$\Delta \bar{w}_{ij}^{L-1}(n) = -\gamma \delta_j^{L-n}(n - (L-1)M) x_i^{L-1-n}(n - (L-1)M) \quad (4)$$

And adaptation starts for $n > (L-1)M$ as pattern index. More details about adaptation of FIR and also adaptive version of FIR can be found in [1,2,3]

4. Correction with Linear subpredictor

The concept of the predictor is clear from figure 1. The overall performance of the prediction is based on ability to extrapolate function and therefore we have to find functional relationship between output and the input. If there is no functional relationship between these 2 elements we have to change the input (the length of the input) to be able to identify functional relation between input and output. In this sense the prediction problem is very similar to classification problem.

The goal of predictor realized by FIR neural network (described in previous section) was to include maximum function dependency between required neural network output and relevant neural network input. In some cases, adaptation process needn't to accomplish optimum and prediction is not complete. We can use error signal and we can obtain better properties of prediction system. There are number of different approaches to use this error signal [2,3,6].

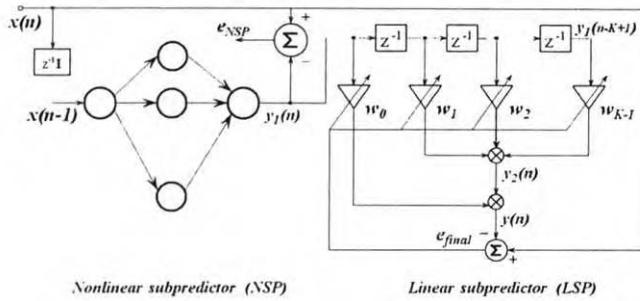


Figure 2 NSP-LSP predictor

One of them is cascade predictor, which consists of two subpredictors (Figure 2):

1. Nonlinear subpredictor (NSP) is multilayer neural network with nonlinear neurons in hidden layer. It used supervised learning and $x(n)$ is desired output. NSP is independent from second subpredictor, which is behind it.
2. Linear subpredictor (LSP) is realized by one FIR synapsis. LSP used supervised learning LMS and output from NSP is input to LSP. Desired output is $x(n)$ too.

Sample $x(n)$ is desired output for both of subpredictors because:

- It is usually very difficult to extract desired output from data only for the linear part of time series. Realization of separating linear part of time series from nonlinear part of time series is complicated.
- Output neuron of NSP can be nonlinear or linear and NSP may partially predict linear part of time series, therefore the difference between the expected output and real output may be smaller.

It is possible to divide this method of cascade prediction into two steps:

1. In the first step nonlinear and partially linear dependencies by NSP predictor are predicted.
2. In the second step, LSP predictor decreases error of prediction.

5. Correction with Nonlinear subpredictor

If there exists an assumption that error signal of neural network is a predictable time series. It is possible to create a model which extrapolates error in the next step following errors in previous steps. Ability to predict errors depends on the part of noise signal included in data and error too. In case that error signal contains, prediction of error is rather difficult. Neural network error signal $e(n-1)$ is input to neural network, which is a predictor of error.

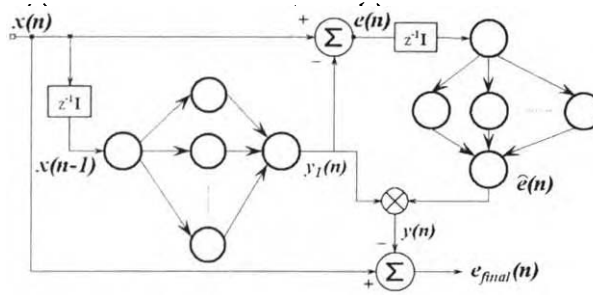


Figure 3 Correction with NSP

This network provided error of time series makes estimate of error $\hat{e}(n)$ that is added to the output $y_i(n)$ in next step. The error of prediction system $e_{final}(n)$ is expected to be less in mean then Error of prediction system without this correction:

$$E[e_{final}] < E[e] \quad (5)$$

where E is statistical operator of mean.

We can divide this method into two steps:

- In the first step to train the first network that extrapolates future trend of time series
- In the second step, approximation made by error of the first network within assumption that error signal is predictable.

6. Experiments and results

The presented testbed is coming from the benchmark and also real/world data. The first data set is from Santa-Fe prediction competition and it is related to magnetic data from the Sun. The second data set is a set of electricity load forecast and the aim is to predict an ultra-short load in the system.

The first set consists of 280 points and it indicates sun spot activity. This set was divided into a training set (221 points) and a testing set (next 59 points). The second set consists of hour samples of electricity load measured in one junction of energetic system. Training set (2000 samples) contains data for 3 months approximately and testing set (500 samples) contains data for the same time for next year. For comparison purposes mean percentage error was used:

$$MAPE = \frac{1}{N} \sum_{i=1}^N |d_i - x_i| \cdot 100\% \quad (6)$$

where x_i is a predicted value d_i is a real value and N is a number of patterns. The second prediction quality parameter is maximal error between computed and real values during prediction testing.

The results of experiments and network topologies are presented in Table 1. It is clear that the second parameter of prediction quality (max. error) has been improved and that this approach provides better prediction results.

Data	Type of correction	No. of neurons in layers	Length of FIR filters	MAPE [%]	MAPE [%] after correction	Maximum absolute error	Maximum absolute error after correction
Sun spot	NSP	1-10-1	10-5	7,22	6,45	0,518	0,462
Energy	LSP	-	10 to 15	3,02	2,93	-	-
	NSP	1-20-1	18-5	3,02	2,18	0,186	0,135

Table 1 Results of experiments

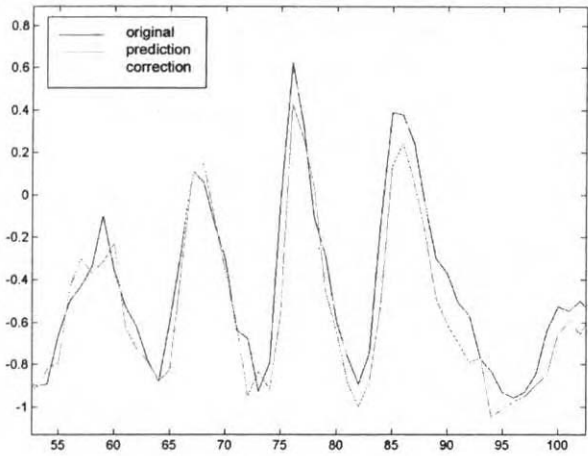


Figure 4 Results of experiments with sun spot data.

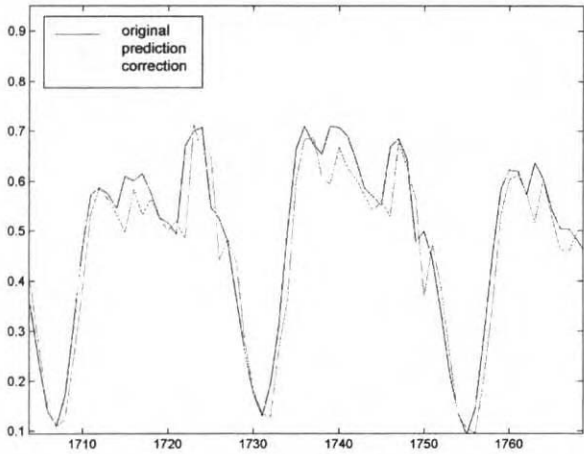


Figure 5 Results of experiments with electricity load forecast data

7. Conclusion

In the paper is presented the approach of prediction improvement using cascade approach of 2 different types of prediction approaches while improving the overall prediction quality. The experimental results indicate improvement in mean prediction error and also in minimalization of maximal prediction error. The latest prediction parameter can be crucial in many applications e.g. load forecast where the maximal error value is large could be very risky to use for these type of prediction problem. If the maximal error is big it could lead to significant financial losses and in some applications also in damages. The relation of maximal and mean error is weakening if the test prediction interval is too long and maximal error is hidden by mean error calculation. For these reasons we consider this approach as useful in prediction systems.

8. References

- [1] Wan. E.A. Finite impulse response neural networks with application in time series prediction. Stanford University, 1993
- [2] Wan. E.A. Finite impulse response neural networks for autoregressive time series prediction. Stanford University, 1993
- [3] K. Nakayama, A. A. M khalaf. A hybrid nonlinear predictor: Analysis of learning process and predictability for noisy time series. Kanazava University, 1999
- [4] Kropuch M., Neural networks in prediction systems, Diploma thesis, TU Košice , 2001
- [5] Wan. E.A. Modeling nonlinear dynamics with neural networks. Stanford University, 1993
- [6] Hykin S. Neural networks – A comprehensive foundation. Macmillian Publishing, 1994
- [7] Novotný D.: Intelligent prediction systems, thesis, TU Kosice 2002
- [8] Fahlman S. E. Leibiére Ch.. The cascade-correlation learning achitecture. Carneige Mellon University, Pittsburgh, 1991
- [9] M. C. Mozer. Neural net architectures for temporal sequence processing. Institute of Cognitive science, University of Colorado, 1993
- [10] A. D. Back, A. Ch. Tsoi. Aspects of adaptive learning algorithms for FIR feed-forward networks. University of Queensland, Brisbane, 1994
- [11] P. Szathmary, M. Kolcun. Daily load electricity forecasting using neural networks. Technical University, Kosice, 2000

Complex Cognitive Phenomena – Challenge for Artificial Intelligence

Marek DOBEŠ

*Institute for Social Sciences, Slovak Academy of Sciences,
Karpatská 5, 04001 Košice, Slovak Republic
dobes@saske.sk*

Abstract. Complex cognitive phenomena like planning or consciousness are still not much debated topic in artificial intelligence. This article provides some simple cues on how mental representations that are prerequisite for higher-order cognitive phenomena like insight or self can be modelled using artificial means. Based on inner models of real-world objects, more complex models of situations as well as of inner workings of a system itself can be realised.

Keywords: cognitive phenomena, mental representation, self, neural networks

1. Introduction and problem definition

Until recently, most of the work in the area of artificial intelligence has stopped at the door of higher-order cognitive phenomena like consciousness, self-concept, etc. It is paradoxical in a way, as memory and learning (that are quite well covered in cognitive modelling research – see for example Levine, 1991) form a continuum with other complex human cognitive phenomena mentioned above. Things are moving on, however, and attempts to define more of higher cognitive phenomena appear (see for example Aleksander, 1996, Sun, 1999).

Objective of this article is to contribute to a theoretical framework for research of more complex cognitive phenomena in the area of computational intelligence (specifically mental representations that then appear universal prerequisite for planning and self-concept – compare Hoyle et al., 1999), and sketch possibilities of modelling such phenomena using neural networks.

2. Mental representations

A mental representation of objects is connected chiefly with associative areas of the brain where outputs of various modalities from sensory areas (visual, auditory, etc.) fuse. In neurophysiological terms it can be defined as a state of activity of defined populations of neurons (Changeux, Dehaene, 1989).

A mental representation differs from a simple inner image of an object. It is more a set of functional properties enabling us to anticipate behaviour (or output) of an object from stimuli that interact with it (inputs).

There are several reasons why inner models of reality are important for human (as well as artificial) intelligence:

They serve as a way to stack knowledge, they enable inner (and therefor quicker and less costly or painful) manipulation with objects – a prerequisite for planning or insight.

An ability to form mental representations is also an important prerequisite for a self-concept of men. Like a functional model of any object, self-concept can be looked at as a model of our own functioning. We behave in a certain way and receive feedback from others that helps us to form an image of how we may look like in the eyes of outside world.

3. Neural network model of a mental representation

To suggest a design of creation of mental representations using neural networks let us introduce following considerations on how it could possibly work. Let me mention a set of components necessary for such a computation.

3.1 Motivation module

Task of motivation module, similarly to biological systems, is to ensure activity of a system until a task is completed (completion of a task then reduces urge to activity – so called drive reduction). Here is a simple example of how such a motivation module could look like:

Let us consider a neurone (set of neurones respectively) that continuously produces some output. We can achieve this for example by adding a bias so, that neurone is constantly activated. Thus, it will bring activity to a system, until a system (after completion of a given condition) inhibits it.

In biological systems inhibition (as well as disinhibition) happens gradually (for example as levels of sugar in blood rise or fall). The same can be done in an artificial system by using a set of neurones with different threshold values.

3.2 Real object

How shall a real object that we are going to model be represented? it should possess some basic properties: It should be able to input stimuli that are sent to it from the environment, compute its response to them, and send its outputs back to the environment. In the artificial environment, it can be a function that computes inputs to outputs.

3.3 Imitation module

Let us consider a neural network that has the same number of inputs and outputs as a real object that we are going to mimic. After a learning process that will be mentioned below, such a network will represent the object we need.

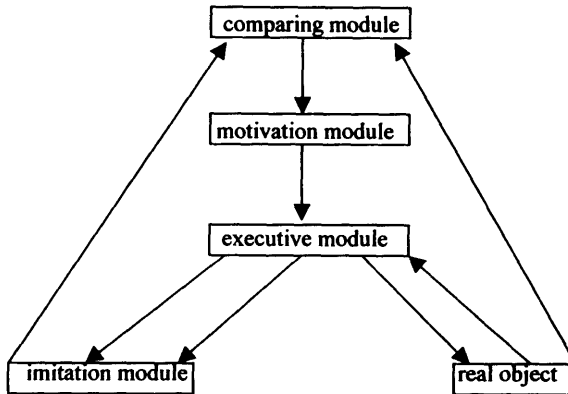
3.4 Executive module

Executive neural network is activated by a motivation module so as to provide stimuli for a real object. Executive module sends a set of more or less random stimuli towards the real object and collects its responses. These random stimuli are also used as inputs to the imitation module. Responses from the real object are used as targets for the training imitation module.

3.5 Comparing module

Another network is used to compare outputs from the real object and those from the imitation module. Its outputs are connected to the motivation module and inhibit it only if responses from real the object and the imitation module are close enough. Only then the process of creation of a model of the real object is finished.

Following scheme shows how modules are interconnected:



Scheme 1. Creation of inner model of an real object.

At the end of this process, the imitation module becomes a representation of the real object. We could design such a network also directly by training it on inputs and outputs of the real object. However, above mentioned system can design it independently and can automatically support more real objects and respective imitation networks.

4. Neural network model of a situation

Similar process can be used if we would like to create an inner model not only of an object but one of a situation. We can regard a situation as an interaction of real objects.

If we want to create a model of a situation, it can be again illustrated using an interaction of modules. A situation can be represented by a function – inputs enter real objects that interact with each other and provide an output. For our system to mimic such behaviour, we need to have inner models of real objects entering the situation. Then our system should work out the way of combining inner models so that for given inputs their interaction provides the same outputs as a real situation.

Let us have a motivation module again, this time, however, it provides us with random sequences in which inner models should interact. This sequence then determines, in what way outputs from one model are chained to form inputs to another one. In this way an output is generated that is compared with the output of a real situation. This repeats until a proper sequence is found. This is a trial-and-error method at first. However, when we introduce a neural network module that correlates the sequence of interaction of inner models and inputs, an internal model of the system's own activity emerges.

5. Conclusion

Existence of models of real objects is a prerequisite for inner manipulation with objects. By modelling complex cognitive phenomena like planning or insight we need to search through a set of inner models, and combining them we attempt to find a sequence that solves a given problem.

Modelling of self also relies on an inner model of workings of the system itself. When a system is complex enough, it is of an advantage to have a model that in less time predicts the reaction of the system to outer stimuli.

References

- [1] ALEKSANDER, I. (1996) Neuroconsciousness: A theoretical framework. *Neurocomputing*, 12, pp. 91-111.
- [2] CHANGEUX, J. P., DEHAENE, S. (1989) Neuronal models of cognitive functions. *Cognition*, 33, pp. 63-109.
- [3] HOYLE, R. H. (1999) *Selfhood*. Westview Press, Oxford.
- [4] LEVINE, D. S. (1991) *Introduction to neural and cognitive modeling*. Lawrence Erlbaum Associates, Hillsdale.
- [5] SUN, R. (1999) Accounting for the computational basis of consciousness: A connectionist approach. *Consciousness and Cognition*, 8, pp. 529-565.

Convergence of Action Dependent Dual Heuristic Dynamic Programming Algorithms in LQ Control Tasks

Dušan Krokavec

Technical University of Košice,

Faculty of Electrical Engineering and Informatics

Department of Cybernetics and Artificial Intelligence

Letná 9/B, 042 00 Košice, Slovak Republic

email:krokavec@ccsun.tuke.sk

Abstract. The purpose of the paper is to present an algorithm to solve the optimization problems concerning with robust LQ control, as well as a method how the exposed problems can be reduced to a standard formulation using steepest-descent gradient function principles, where dual heuristic dynamic programming is used. Robust LQ control design is specific by more general one-step cost function, than ones usually used for critic-based training, and with greedy minimization for updating not only the action but also the critic.

Keywords: *Adaptive critic design, Dual heuristic programming, Neural networks, LQ control*

1 Introduction

In general, optimization problem solving need methods that can capture high order complexity and uncertainty of the systems. One class of this methods is adaptive critic design. This method can be considered as approximations of dynamic programming using neural networks. A typical adaptive critic design includes action, critic and model modules, where the model module, in general, simulates a subject to control. Each module can be a neural network or, alternatively, any differentiable system.

Heuristic dynamic programming is a neural network approach to solve the Bellman equation and a good knowledge of derivatives of an optimization criterion is a prerequisite to find a satisfactory solution. Dual Heuristic Dynamic Programming have an important advantage since its critic module produces a representation for derivatives being explicitly trained on them through the derivatives of criterion and differentiable model of system.

The paper studies dual heuristic dynamic programming convergence in the domain of well known problem of linear quadratic control (LQ) starting with Q-learning. Since robust LQ control design is specific by more general one-step cost function than ones usually used for critic-based training, and with greedy minimization for updating not only the action but also the critic, the known convergence results and their limitation [1], [6] are generalized.

2 Discrete-time Robust LQ Control

In general, a discrete-time multi-variable uncertain linear system can be considered as

$$\mathbf{x}(i+1) = (\mathbf{F} + \Delta\mathbf{F}(i))\mathbf{x}(i) + (\mathbf{G} + \Delta\mathbf{G}(i))\mathbf{u}(i) , \quad (1)$$

$$\mathbf{y}(i) = \mathbf{C}\mathbf{x}(i) , \quad (2)$$

vectors $\mathbf{x}(i) \in \mathbb{R}^n$, $\mathbf{u}(i) \in \mathbb{R}^r$, $\mathbf{y}(i) \in \mathbb{R}^m$ and matrices $\mathbf{F} \in \mathbb{R}^{n \times n}$, $\mathbf{G} \in \mathbb{R}^{n \times r}$, $\mathbf{C} \in \mathbb{R}^{m \times n}$ are finite valued, and $\Delta\mathbf{F}(i) \in \mathbb{R}^{n \times n}$, $\Delta\mathbf{G}(i) \in \mathbb{R}^{n \times r}$ are unknown matrices which represent time-varying parametric uncertainties. It is assumed that considered uncertainty matrices to be of the form

$$[\Delta\mathbf{F}(i) \ \Delta\mathbf{G}(i)] = \mathbf{M}\mathbf{H}(i) [\mathbf{N}_1 \ \mathbf{N}_2] , \quad (3)$$

$$\mathbf{H}^T(i)\mathbf{H}(i) \leq \mathbf{I} , \quad (4)$$

where $\mathbf{N}_1 \in \mathbb{R}^{h \times n}$, $\mathbf{N}_2 \in \mathbb{R}^{h \times r}$, $\mathbf{M} \in \mathbb{R}^{n \times h}$.

For the controllable discrete-time multi-variable system (1), (2) the optimal nominal control design task is to determine the control

$$\mathbf{u}(i) = -\mathbf{K}(i)\mathbf{x}(i) , \quad (5)$$

that minimizes the quadratic cost function

$$J_N = \sum_{i=0}^{\infty} (\mathbf{x}^T(i)\mathbf{Q}_0\mathbf{x}(i) + \mathbf{x}^T(i)\mathbf{S}_0\mathbf{u}(i) + \mathbf{u}^T(i)\mathbf{S}_0^T\mathbf{x}(i) + \mathbf{u}^T(i)\mathbf{R}_0\mathbf{u}(i)) , \quad (6)$$

where $\mathbf{Q}_0 \in \mathbb{R}^{n \times n}$ is a symmetric positive semi-definite matrix, $\mathbf{R}_0 \in \mathbb{R}^{m \times m}$ is a symmetric positive definite matrix, and $\mathbf{K}(i) \in \mathbb{R}^{n \times r}$ is the optimal control gain matrix.

Parameter uncertainty (3), (4) can be accepted using its upper bounds [4] in the form of equivalent matrices (with $0 < \varepsilon \ll 1$)

$$\mathbf{Q}^* = \frac{1}{\varepsilon}\mathbf{N}_1\mathbf{N}_1^T, \quad \mathbf{R}^* = \frac{1}{\varepsilon}\mathbf{N}_2\mathbf{N}_2^T, \quad \mathbf{S}^{*T} = \frac{1}{\varepsilon}\mathbf{N}_1\mathbf{N}_2^T, \quad \mathbf{M}^* = \varepsilon\mathbf{M}^T\mathbf{M} , \quad (7)$$

which is equivalent to modification of the performance index (6)

$$J_R = \sum_{i=0}^{\infty} (\mathbf{x}^T(i)\mathbf{Q}\mathbf{x}(i) + \mathbf{x}^T(i)\mathbf{S}\mathbf{u}(i) + \mathbf{u}^T(i)\mathbf{S}^T\mathbf{x}(i) + \mathbf{u}^T(i)\mathbf{R}\mathbf{u}(i)) , \quad (8)$$

where

$$\mathbf{Q} = \mathbf{Q}_0 + \mathbf{Q}^*, \mathbf{R} = \mathbf{R}_0 + \mathbf{R}^*, \mathbf{S} = \mathbf{S}_0 + \mathbf{S}^* . \quad (9)$$

An optimal trajectory (5) is generated for state feedback gain matrix $\mathbf{K} \in \mathbb{R}^{n \times r}$, which is the steady-state solution of the gain matrix

$$\mathbf{K}(i) = (\mathbf{R} + \mathbf{G}^T\mathbf{P}(i)\mathbf{G})^{-1}(\mathbf{S}^T + \mathbf{G}^T\mathbf{P}(i)\mathbf{F}) \quad (10)$$

and discrete Riccati equation

$$\begin{aligned} \mathbf{P}(i-1) &= \mathbf{Q} + \mathbf{F}^T\mathbf{P}(i)\mathbf{F} - (\mathbf{S} + \mathbf{F}^T\mathbf{P}(i)\mathbf{F})\mathbf{K}(i) = \\ &= \mathbf{Q} + \mathbf{F}^T\mathbf{P}(i)\mathbf{F} + (\mathbf{S} + \mathbf{F}^T\mathbf{P}(i)\mathbf{F})(\mathbf{R} + \mathbf{G}^T\mathbf{P}(i)\mathbf{G})^{-1}(\mathbf{S}^T + \mathbf{G}^T\mathbf{P}(i)\mathbf{F}) . \end{aligned} \quad (11)$$

The symmetric matrix \mathbf{P} is then a nonnegative-definite steady-state solution of the algebraic matrix equation determined by (11).

3 Parametrization of LQ Control Task

The above mentioned problem is equivalent to finding the Ljapunov function $V(\mathbf{x}(i))$ of special structure to system (1), (2). Assuming that

$$f(\mathbf{x}(i), \mathbf{u}(i)) = \mathbf{F}\mathbf{x}(i) + \mathbf{G}\mathbf{u}(i) = \mathbf{u}(i+1) , \quad (12)$$

$$r(\mathbf{x}(i), \mathbf{u}(i)) = \mathbf{x}^T(i)\mathbf{Q}\mathbf{x}(i) + \mathbf{u}^T(i)\mathbf{R}\mathbf{u}(i) + \mathbf{x}^T(i)\mathbf{S}\mathbf{u}(i) + \mathbf{u}^T(i)\mathbf{S}^T\mathbf{x}(i) , \quad (13)$$

$$g(\mathbf{x}(i)) = -\mathbf{K}(i)\mathbf{x}(i) = \mathbf{u}(i) , \quad (14)$$

$$V(\mathbf{x}(i)) = \mathbf{x}^T(i)\mathbf{P}(i-1)\mathbf{x}(i) , \quad (15)$$

$$\Delta V(\mathbf{x}(i)) = \mathbf{x}^T(i+1)\mathbf{P}(i)\mathbf{x}(i+1) - \mathbf{x}^T(i)\mathbf{P}(i-1)\mathbf{x}(i) , \quad (16)$$

the q function can be defined as

$$\begin{aligned} q(\mathbf{x}(i), \mathbf{u}(i)) &= r(\mathbf{x}(i), \mathbf{u}(i)) + \Delta V(\mathbf{x}(i)) = \\ &= \mathbf{x}^T(i+1)\mathbf{P}(i)\mathbf{x}(i+1) - \mathbf{x}^T(i)\mathbf{P}(i-1)\mathbf{x}(i) + \\ &+ \mathbf{x}^T(i)\mathbf{Q}\mathbf{x}(i) + \mathbf{x}^T(i)\mathbf{S}\mathbf{u}(i) + \mathbf{u}^T(i)\mathbf{S}^T\mathbf{x}(i) + \mathbf{u}^T(i)\mathbf{R}\mathbf{u}(i) = \\ &= [\mathbf{x}^T(i), \mathbf{u}^T(i)] \left(\begin{bmatrix} \mathbf{Q} - \mathbf{P}(i-1) & \mathbf{S} \\ \mathbf{S}^T & \mathbf{R} \end{bmatrix} + \begin{bmatrix} \mathbf{F}^T \\ \mathbf{G}^T \end{bmatrix} \mathbf{P}(i) \begin{bmatrix} \mathbf{F} & \mathbf{G} \end{bmatrix} \right) \begin{bmatrix} \mathbf{x}(i) \\ \mathbf{u}(i) \end{bmatrix} . \end{aligned} \quad (17)$$

Equation (17) may be written as

$$q(\mathbf{p}(i)) = \mathbf{p}^T(i)\mathbf{L}(i)\mathbf{p}(i) = \mathbf{p}^T(i) \begin{bmatrix} \mathbf{L}_{11}(i) & \mathbf{L}_{12}(i) \\ \mathbf{L}_{21}(i) & \mathbf{L}_{22}(i) \end{bmatrix} \mathbf{p}(i) , \quad (18)$$

where

$$\mathbf{L}(i) = \begin{bmatrix} \mathbf{L}_{11}(i) & \mathbf{L}_{12}(i) \\ \mathbf{L}_{21}(i) & \mathbf{L}_{22}(i) \end{bmatrix} = \begin{bmatrix} \mathbf{Q} + \mathbf{F}^T\mathbf{P}(i)\mathbf{F} - \mathbf{P}(i-1) & \mathbf{S} + \mathbf{F}^T\mathbf{P}(i)\mathbf{G} \\ \mathbf{S}^T + \mathbf{G}^T\mathbf{P}(i)\mathbf{F} & \mathbf{R} + \mathbf{G}^T\mathbf{P}(i)\mathbf{G} \end{bmatrix} , \quad (19)$$

$$\mathbf{p}(i) = \begin{bmatrix} \mathbf{x}(i) & \mathbf{u}(i) \end{bmatrix}^T . \quad (20)$$

This implies, that

$$\frac{\partial q(\mathbf{p}(i))}{\partial \mathbf{u}(i)} = \mathbf{p}^T(i) \begin{bmatrix} \mathbf{L}_{11}(i) & \mathbf{L}_{12}(i) \\ \mathbf{L}_{21}(i) & \mathbf{L}_{22}(i) \end{bmatrix} \begin{bmatrix} \mathbf{0} \\ \mathbf{1} \end{bmatrix} = [\mathbf{x}^T(i) \mathbf{u}^T(i)] \begin{bmatrix} \mathbf{L}_{12}(i) \\ \mathbf{L}_{22}(i) \end{bmatrix} = \mathbf{0} , \quad (21)$$

and the optimal control law is

$$\mathbf{u}(i) = -\mathbf{L}_{22}^{-T}(i)\mathbf{L}_{12}^T(i)\mathbf{x}(i) = (\mathbf{R} + \mathbf{G}^T\mathbf{P}(i)\mathbf{G})^{-1}(\mathbf{S}^T + \mathbf{G}^T\mathbf{P}(i)\mathbf{F})\mathbf{x}(i) . \quad (22)$$

Correspondingly, (19) reduces to the matrix Riccati equation

$$\frac{\partial q(\mathbf{p}(i))}{\partial \mathbf{x}(i)} = \mathbf{p}^T(i) \begin{bmatrix} \mathbf{L}_{11}(i) & \mathbf{L}_{12}(i) \\ \mathbf{L}_{21}(i) & \mathbf{L}_{22}(i) \end{bmatrix} \begin{bmatrix} \mathbf{1} \\ \mathbf{0} \end{bmatrix} = [\mathbf{x}^T(i) \mathbf{u}^T(i)] \begin{bmatrix} \mathbf{L}_{11}(i) \\ \mathbf{L}_{21}(i) \end{bmatrix} = \mathbf{0} , \quad (23)$$

and

$$\begin{aligned} \mathbf{P}(i-1) &= \\ &= \mathbf{Q} + \mathbf{F}^T\mathbf{P}(i)\mathbf{F} + (\mathbf{S} + \mathbf{F}^T\mathbf{P}(i)\mathbf{G})(\mathbf{R} + \mathbf{G}^T\mathbf{P}(i)\mathbf{G})^{-1}(\mathbf{S}^T + \mathbf{G}^T\mathbf{P}(i)\mathbf{F}) . \end{aligned} \quad (24)$$

4 Action Neural Network

One way to search for the optimal parameters is to employ a gradient algorithm. In this case the requirement is to calculate derivatives

$$\frac{\partial q(\mathbf{p}(i))}{\partial \mathbf{K}(i)} = \left(\frac{\partial r(\mathbf{x}(i), \mathbf{u}(i))}{\partial g(\mathbf{x}(i))} + \frac{\partial V(\mathbf{x}(i+1))}{\partial \mathbf{x}(i+1)} \frac{\partial f(\mathbf{x}(i), \mathbf{u}(i))}{\partial g(\mathbf{e}(i))} \right) \frac{\partial g(\mathbf{x}(i))}{\partial \mathbf{K}(i)}, \quad (25)$$

since

$$\frac{\partial V(\mathbf{x}(i))}{\partial g(\mathbf{x}(i))} = 0. \quad (26)$$

Thus, if there are no bounds on $\mathbf{u}(i)$, the minimizing $\mathbf{u}(i)$ must be such, that

$$\frac{\partial q(\mathbf{x}(i), \mathbf{u}(i))}{\partial g(\mathbf{x}(i))} = \frac{\partial r(\mathbf{x}(i), \mathbf{u}(i))}{\partial \mathbf{u}(i)} + \frac{\partial V(\mathbf{x}(i+1))}{\partial \mathbf{x}(i+1)} \frac{\partial f(\mathbf{x}(i), \mathbf{u}(i))}{\partial \mathbf{u}(i)} = 0, \quad (27)$$

i.e. all desired output values of action network are defined as zeros, and the target for an action network minimization is

$$\mathbf{e}_a(i) = \mathbf{0} - \frac{\partial q(\mathbf{x}(i), \mathbf{u}(i))}{\partial g(\mathbf{x}(i))}. \quad (28)$$

With criterion

$$W_a(i) = \mathbf{e}_a^T(i) \mathbf{e}_a(i) = \sum_{h=1}^M e_{ah}^2(i) \quad (29)$$

for the action network training, the steepest-descent discrete gradient method, using error back-propagation algorithm, can be applied to synaptic weight minimization as

$$\begin{aligned} \Delta w_{rs}(i) &= -\mu_a \frac{\partial W_a(i)}{\partial w_{rs}(i)} = \\ &= -\mu_a \sum_{k=1}^M \left(\frac{\partial r(i)}{\partial u_k(i)} + \sum_{j=1}^N \frac{\partial V(i+1)}{\partial x_j(i+1)} \frac{\partial x_j(i+1)}{\partial u_k(i)} \right) \frac{\partial u_k(i)}{\partial w_{rs}(i)}, \end{aligned} \quad (30)$$

where

- $\frac{\partial e_j(i+1)}{\partial u_k(i)}$ is calculated from analytical equation of the system model,
- $\frac{\partial V(i+1)}{\partial e_j(i+1)}$ may be approximated by the critic,
- $\frac{\partial r(i)}{\partial u_k(i)}$ is calculated as a derivative of criterion utility function,

and N, M is the number of state and input variables, respectively.

5 Critic Neural Network

Obviously, (23) define the general conditions for critic neural network optimization. Direct application of (23) results the next target for a critic network minimization

$$e_c^*(i) = 0 - \frac{\partial q(\mathbf{x}(i), \mathbf{u}(i))}{\partial \mathbf{x}(i)}, \quad (31)$$

with backward in-time training of hidden critic network parameters.

Another way of reformulating the condition (23) is to replace them with the sufficient condition (dual dynamic programming principle)

$$\frac{\partial q(\mathbf{p}(i))}{\partial \mathbf{x}(i)} = \frac{\partial r(\mathbf{x}(i), \mathbf{u}(i))}{\partial \mathbf{x}(i)} + \frac{\partial V(\mathbf{x}(i+1))}{\partial \mathbf{x}(i+1)} \frac{\partial f(\mathbf{x}(i), \mathbf{u}(i))}{\partial \mathbf{x}(i)} - \frac{\partial V(\mathbf{x}(i))}{\partial \mathbf{x}(i)} = 0. \quad (32)$$

Critic network estimates the derivatives of Ljapunov function with respect to the system state vector $\mathbf{x}(i)$, which gives for the j th desired output of the critic neural network

$$\begin{aligned} c_j^0(i) &= \frac{\partial V(i)}{\partial x_j(i)} = \frac{r(\mathbf{x}(i), \mathbf{u}(i))}{\partial x_j(i)} + \frac{r(\mathbf{x}(i), \mathbf{u}(i))}{\partial u(i)} \frac{\partial u(i)}{\partial x_j(i)} + \frac{V(\mathbf{x}(i+1))}{\partial \mathbf{x}(i+1)} \frac{\partial \mathbf{x}(i+1)}{\partial x_j(i)} = \\ &= \frac{r(i)}{\partial x_j(i)} + \sum_{k=1}^M \frac{\partial r(i)}{\partial u_k(i)} \frac{\partial u_k(i)}{\partial x_j(i)} + \sum_{h=1}^N \frac{\partial V(i+1)}{\partial x_h(i+1)} \frac{\partial x_h(i+1)}{\partial x_j(i)} + \\ &\quad + \sum_{k=1}^M \sum_{h=1}^N \frac{\partial V(i+1)}{\partial x_h(i+1)} \frac{\partial x_h(i+1)}{\partial u_k(i)} \frac{\partial u_k(i)}{\partial x_j(i)}, \end{aligned} \quad (33)$$

where

- $\frac{\partial x_h(i+1)}{\partial u_k(i)}$ is calculated from analytical equation of the system model,
- $\frac{\partial V(i+1)}{\partial x_h(i+1)}$ is approximated by the critic,
- $\frac{r(i)}{\partial x_j(i)}$ are calculated as a derivative of criterion utility function,
- $\frac{\partial r(i)}{\partial u_k(i)}$ are calculated as a derivative of criterion utility function,
- $\frac{\partial u_k(i)}{\partial x_j(i)}$ is given as the product of synaptic weights on the path from j th input to k th output of action neural network, respectively.

The critic network tries to minimize the following error measure over time

$$W_c = \sum_{i=1}^{\infty} W_c(i) = \sum_{i=1}^{\infty} (c(i) - c^0(i))^2, \quad (34)$$

where desired output of the critic neural network in the time-point i is (33), i.e. the output of the critic neural network in the last-but-one step of iteration is used as desired ones. The critic network is trained forward in time, which is of the great importance for real-time operation.

The training criterion for critic neural network can be defined as

$$W_c(i) = \frac{1}{2} \sum_{i=1}^N (c_j(i) - c_j^0(i))^2, \quad (35)$$

and the optimization procedure is given by

$$\Delta w_{rs}(i) = -\mu_c \frac{\partial W_c(i)}{\partial w_{rs}(i)} . \quad (36)$$

Desired output of the critic network is typically calculated by running the critic network one more computational cycle to provide its next-in-time output, and then use this value to compute the target for the present-time cycle. Since the critic network that calculates the target is changing with each update, it provides a moving target for critic neural network training.

6 Convergence of Learning Algorithm

While dynamic programming is used for estimation of a function $V(\mathbf{x}(i))$, dual dynamic programming is a method for estimation of $\partial V(\mathbf{x}(i))/\partial \mathbf{x}(i)$, rather than $V(\mathbf{x}(i))$ itself. Assuming, that (12) - (16) are given, then (15), (33) imply

$$\frac{\partial V(\mathbf{x}(i))}{\partial \mathbf{x}(i)} = \mathbf{x}^T(i) \mathbf{P}(i-1) , \quad (37)$$

$$d(i) = \frac{\partial}{\partial \mathbf{x}(i)} (r(\mathbf{x}(i), \mathbf{u}(i)) + V(\mathbf{x}(i+1))) = \mathbf{x}^T(i) \mathbf{J}(i) , \quad (38)$$

$$\begin{aligned} \mathbf{J}(i) = & \mathbf{Q} - \mathbf{S} \mathbf{K}(i) - \\ & - \mathbf{K}^T(i) \mathbf{S}^T + \mathbf{K}^T(i) \mathbf{R} \mathbf{K}(i) + (\mathbf{F} - \mathbf{G} \mathbf{K}(i))^T \mathbf{P}(i) (\mathbf{F} - \mathbf{G} \mathbf{K}(i)) . \end{aligned} \quad (39)$$

Equation (37) cannot be applied for any supervised learning algorithm that tries to approximate gradient of $V(\mathbf{x}(i))$ using the steepest-descent algorithm (36) since algorithm divergence. To solve the minimization problem with respect to $\mathbf{P}(i)$ the next form of predicted value of (37) is used

$$\frac{\partial V(\mathbf{x}(i))}{\partial \mathbf{x}(i)} = \mathbf{x}^T(i) \mathbf{P}(i) , \quad (40)$$

and $W_c(i)$ is given as below

$$W_c(i) = |\mathbf{x}^T(i) \mathbf{P}(i) - d(i)|^2 = |(\mathbf{P}(i) \mathbf{x}(i) - d(i))^T|^2 . \quad (41)$$

Thus,

$$\mathbf{P}(i+1) = \mathbf{P}(i) - \mu \frac{\partial W_c(i)}{\partial \mathbf{x}(i)} = \mathbf{P}(i) - \mu \mathbf{X}(i) (\mathbf{P}(i) - \mathbf{J}(i)) \quad (42)$$

can be used to solve the minimization problem, where μ is a positive learning parameter determining the speed of convergence and $\mathbf{X}(i) = \mathbf{x}(i) \mathbf{x}^T(i)$ is a symmetric matrix, which modifies the speed of convergence. For $\mathbf{X}(i) = \mathbf{I}$

$$\mathbf{P}(i+1) = \mathbf{P}(i) - \mu (\mathbf{P}(i) - \mathbf{J}(i)) . \quad (43)$$

Equation (43) represents resulting approximation of critic values for given training set where $\mathbf{X}(i) = \mathbf{I}$ and \mathbf{I} is the identity matrix.

Theorem 1. Let $\mu \in (0, 1)$, $\mathbf{R} > \mathbf{0}$, $\mathbf{Q} \geq \mathbf{0}$ and (\mathbf{F}, \mathbf{G}) is a controllable pair. There exists a matrix $\mathbf{P}^* \geq \mathbf{0}$ such that $\mathbf{P}(i) \rightarrow \mathbf{P}^*$, where $\mathbf{P}(i)$ are the elements of the sequence $\{\mathbf{P}(i) : i = 0, 2, \dots\}$ given by $\mathbf{P}(i+1) = \mathbf{P}(i) - \mu(\mathbf{P}(i) - \mathbf{J}(i))$ for $\mathbf{J}(i)$ as in (39).

Proof. Let there are two sequences $\{\mathbf{P}(j) : j = i, i+1, \dots\}$, $\{\mathbf{P}^*(j) : j = i, i+1, \dots\}$ starting from the same starting value $\mathbf{P}(i) = \mathbf{P}^*(i)$ and

$$\mathbf{J}(i) = \mathbf{Q} - \mathbf{S}\mathbf{K}(i) - \mathbf{K}^T(i)\mathbf{S}^T + \mathbf{K}^T(i)\mathbf{R}\mathbf{K}(i) + (\mathbf{F} - \mathbf{G}\mathbf{K}(i))^T\mathbf{P}(i)(\mathbf{F} - \mathbf{G}\mathbf{K}(i)) , \quad (44)$$

$$\mathbf{J}^*(i) = \mathbf{Q} - \mathbf{S}\mathbf{K}^*(i) - \mathbf{K}^{*T}(i)\mathbf{S}^T + \mathbf{K}^{*T}(i)\mathbf{R}\mathbf{K}^*(i) + (\mathbf{F} - \mathbf{G}\mathbf{K}^*(i))^T\mathbf{P}^*(i)(\mathbf{F} - \mathbf{G}\mathbf{K}^*(i)) \quad (45)$$

Since (24) implies

$$\mathbf{K}^*(i) = (\mathbf{R} + \mathbf{G}^T\mathbf{P}^*(i)\mathbf{G})^{-1}(\mathbf{S}^T + \mathbf{G}^T\mathbf{P}^*(i)\mathbf{F}) , \quad (46)$$

then

$$\begin{aligned} \mathbf{J}^*(i) &= \mathbf{J}(i) + \mathbf{O}(i) = \\ &= \mathbf{Q} - \mathbf{S}(\mathbf{K}(i) + \mathbf{K}^*(i) - \mathbf{K}(i)) - (\mathbf{K}(i) + \mathbf{K}^*(i) - \mathbf{K}(i))^T\mathbf{S}^T + \\ &\quad + (\mathbf{K}(i) + \mathbf{K}^*(i) - \mathbf{K}(i))^T\mathbf{R}(\mathbf{K}(i) + \mathbf{K}^*(i) - \mathbf{K}(i)) + \\ &\quad + (\mathbf{F} - \mathbf{G}(\mathbf{K}(i) + \mathbf{K}^*(i) - \mathbf{K}(i)))^T\mathbf{P}^*(i)(\mathbf{F} - \mathbf{G}(\mathbf{K}(i) + \mathbf{K}^*(i) - \mathbf{K}(i))) = \\ &= \mathbf{Q} - \mathbf{S}\mathbf{K}(i) - \mathbf{K}^T(i)\mathbf{S}^T + \\ &\quad + \mathbf{K}^T(i)\mathbf{R}\mathbf{K}(i) + (\mathbf{F} - \mathbf{G}\mathbf{K}(i))^T\mathbf{P}^*(i)(\mathbf{F} - \mathbf{G}\mathbf{K}(i)) + \\ &\quad + (\mathbf{K}^*(i) - \mathbf{K}(i))^T((\mathbf{R} + \mathbf{G}^T\mathbf{P}^*(i)\mathbf{G})\mathbf{K}(i) - \mathbf{G}^T\mathbf{P}^*(i)\mathbf{F} - \mathbf{S}^T) + \\ &\quad + ((\mathbf{R} + \mathbf{G}^T\mathbf{P}^*(i)\mathbf{G})\mathbf{K}(i) - \mathbf{G}^T\mathbf{P}^*(i)\mathbf{F} - \mathbf{S}^T)^T(\mathbf{K}^*(i) - \mathbf{K}(i)) + \\ &\quad + (\mathbf{K}^*(i) - \mathbf{K}(i))^T(\mathbf{R} + \mathbf{G}^T\mathbf{P}^*(i)\mathbf{G})(\mathbf{K}^*(i) - \mathbf{K}(i)) = \\ &= \mathbf{J}(i) + [\mathbf{K}^*(i) - \mathbf{K}(i)]^T(\mathbf{R} + \mathbf{G}^T\mathbf{P}^*(i)\mathbf{G})(\mathbf{K}^*(i) - \mathbf{K}(i)) . \end{aligned} \quad (47)$$

Thus

$$\begin{aligned} \mathbf{P}^*(i+1) - \mathbf{P}(i+1) &= \\ &= \mathbf{P}^*(i) - \mathbf{P}(i) - \mu[\mathbf{P}^*(i) - \mathbf{J}^*(i) - \mathbf{P}(i) + \mathbf{J}(i)] = \\ &= (1 - \mu)(\mathbf{P}^*(i) - \mathbf{P}(i)) - \mu\mathbf{O}(i) , \end{aligned} \quad (48)$$

$$\begin{aligned} \mathbf{P}(i+1) - \mathbf{P}^*(i+1) &= (1 - \mu)(\mathbf{P}(i) - \mathbf{P}^*(i)) - \\ &\quad - \mu(\mathbf{F} - \mathbf{G}\mathbf{K}(i))^T(\mathbf{P}(i) - \mathbf{P}^*(i))(\mathbf{F} - \mathbf{G}\mathbf{K}(i)) . \end{aligned} \quad (49)$$

Identity (48) implies that the sequence $\{\mathbf{P}(j) : j = i-1, i, \dots\}$ is non-increasing. The vector form of (49) is

$$\begin{aligned} \text{vec}(\mathbf{P}(i+1) - \mathbf{P}^*(i+1)) &= \\ &= ((1 - \mu)\mathbf{I} - \mu(\mathbf{F} - \mathbf{G}\mathbf{K}(i))^T \otimes (\mathbf{F} - \mathbf{G}\mathbf{K}(i)))\text{vec}(\mathbf{P}(i) - \mathbf{P}^*(i)) . \end{aligned} \quad (50)$$

Since (\mathbf{F}, \mathbf{G}) is a controllable pair there is a matrix $\mathbf{K}(i)$ such that the matrix $(\mathbf{F} - \mathbf{G}\mathbf{K}(i))$ has all its eigenvalues in the open units disc. Since this eigenvalues $\{r_h : h = 1, 2, \dots, n\}$ satisfy $|r_h| < 1$ and the eigenvalues of $(\mathbf{F} - \mathbf{G}\mathbf{K}(i))^T \otimes (\mathbf{F} - \mathbf{G}\mathbf{K}(i))$ are given by $\{r_h^2 :$

$h = 1, 2, \dots, n$ the eigenvalues of the matrix $((1 - \mu)\mathbf{I} - \mu(\mathbf{F} - \mathbf{GK}(i))^T \otimes (\mathbf{F} - \mathbf{GK}(i)))$ satisfy condition

$$|q_h| = |(1 - \mu) - \mu r_h^2| < 1, \quad (51)$$

where $\mu \in (0, 1)$. Hence, there exists a norm in which

$$\| (1 - \mu)\mathbf{I} - \mu(\mathbf{F} - \mathbf{GK}(i))^T \otimes (\mathbf{F} - \mathbf{GK}(i)) \| < 1, \quad (52)$$

the sequence $\{\mathbf{P}(j) : j = i, i + 1, \dots\}$ has an upper bound and therefore has a limit \mathbf{P}^* . Taking limits in (44) one can verify that it is the solution of equations

$$\mathbf{P}^* = \mathbf{Q} - \mathbf{SK}^* - \mathbf{K}^{*T}\mathbf{S}^T + \mathbf{K}^{*T}\mathbf{R}\mathbf{K}^* + (\mathbf{F} - \mathbf{GK}^*)^T\mathbf{P}^*(\mathbf{F} - \mathbf{GK}^*), \quad (53)$$

$$\mathbf{K}^* = (\mathbf{R} + \mathbf{G}^T\mathbf{P}^*\mathbf{G})^{-1}(\mathbf{S}^T + \mathbf{G}^T\mathbf{P}^*\mathbf{F}). \quad (54)$$

Since (53) has a unique positive semi-definite solution and $\mathbf{Q} \geq 0$ the matrix \mathbf{P}^* must be the unique solution. \square

7 Conclusion

The paper describes a framework for neural network application in system stability evaluation and control. The purpose was to demonstrate whether the proposed method is able to achieve the system control quality. The tests were conducted for a power system model to present a neural network technique for stabilizing power system transient process, where single machine infinite bus power system [5] was studied. The system LQ control was simulated using an action neural network and a critic neural network, where nonlinear analytical equations were used to compute system model outputs and the utility function in the training stage. Tests have showed impressive generalization capability of the controllers designed via presented heuristic process. It is obvious from (8) that for robust control more complex utility function need to be used to achieve system robust stability. It turns out that even matrices \mathbf{F} , \mathbf{G} , \mathbf{Q} , \mathbf{S} , \mathbf{R} are not available it is possible to recover the optimal control matrix $\mathbf{K}(i)$ as a result of the algorithm convergence for nominal working condition of the system. Using this procedure one can obtain unknown parameter uncertainty upper bounds as a difference to the nominal system model results. The presented method gives some new focus on application of the computational intelligence method in the nonlinear and robust control.

Acknowledgement

The work presented in the paper was supported by Grant Agency of Ministry of Education and Academy of Science of Slovak Republic VEGA under Grant No. 1/9028/02.

References

- [1] Bradtke, S.J., Ydstie, B.E., Barto, A.G.: Adaptive linear quadratic control using policy iteration. Proceedings of the American Control Conference. Baltimore (1994) 3475-3479

- [2] Filasová, A.: Robust Control Design : An Optimal Control Approach. In: Rudas, I.J., Madarász, L. (eds.): INES '99. Proceedings of the 3rd IEEE Conference on Intelligent Engineering Systems, Stará Lesná, Slovakia. Elfa, Košice (1999) 515–518
- [3] Filasová, A., Kašpříšín, J.: Adaptive Critic Design for Dynamic System Control. *AT& P Journal Plus* **8** (2001) 15–17 (in Slovak)
- [4] Krokavec, D., Filasová, A.: Optimal Stochastic Systems. Elfa, Košice, Slovak Republic (1999)(in Slovak)
- [5] Krokavec, D., Filasová, A.: Application of Heuristic Programming to Dynamic System Stabilization. In: Sinčák, P., Vaščák, J., Kvasnička, V., Mesiar, R. (eds.). *The State of the Art in Computational Intelligence. Advances in Soft Computing*. Physica-Verlag, Heidelberg New York (2000) 68–73
- [6] Landelius, T.: Reinforcement Learning and Distributed Local Model Synthesis. Ph.D. Thesis. Linköping University, Sweden (1997)
- [7] Prokhorov, D.V., Santiago, R.A., Wunsch, D.C.: Adaptive critic design: A case study for neurocontrol. *Neural Networks*, **8** (1995) 1367–1372

II. Fuzzy Systems

This page intentionally left blank

A Prototype-Centered Approach to Adding Deduction Capability to Search Engines -- The Concept of Protoform

Lotfi A. Zadeh *

*Professor in the Graduate School and director, Berkeley initiative in Soft Computing (BISC),
Computer Science Division and the Electronics Research Laboratory, Department of EECS,
University of California, Berkeley, CA 94720-1776;
Telephone: 510-642-4959; Fax: 510-642-1712
E-Mail: zadeh@cs.berkeley.edu*

Existing search engines have many remarkable capabilities. But what is not among them is deduction capability -- the capability to answer a query by drawing on information which resides in various parts of the knowledge base or is augmented by the user. An example of a simple query which cannot be answered by any existing search engine is: "How many UC Berkeley alumni were born in California? " In this case, the query and the query-relevant information are crisp. In the query "How many UC Berkeley alumni are wealthy?" the query-relevant information is crisp but the query is fuzzy. The problem-- which is not widely recognized-- is that much of the information in the knowledge base of a search engine is perception-based. Methods based on bivalent logic and standard probability theory lack capability to operate on perception-based information.

Limited progress toward a realization of deduction capability is achievable through application of methods based on bivalent logic and standard probability theory. But to move beyond the reach of standard methods it is necessary to change direction. In the approach which is outlined, a concept which plays a pivotal role is that of a prototype -- a concept which has a position of centrality in human reasoning, recognition, search and decision processes.

Informally, a prototype may be defined as a sigma-summary, that is, a summary of summaries. With this definition as the point of departure, a prototypical form, or protoform, for short, is defined as an abstracted prototype. As a simple example, the protoform of the proposition "Most Swedes are tall" is "QA's are B's," where Q is a fuzzy quantifier, and A and B are labels of fuzzy sets. Similarly, the protoform of "Usually Robert returns from work at about 6 pm," is "Prob (A) is B," where A is a fuzzy event and B is a fuzzy probability.

Abstraction has levels, just as summarization does. For example, in the case of "Most Swedes are tall," successive abstracted forms are "Most A's are tall," "Most A's are B's" and "QA's are B's."

At a specified level of abstraction, propositions are PF-equivalent if they have identical protoforms. For example, propositions "Usually Robert returns from work at about 6 pm" and "In winter, the average daily temperature in Berkeley is usually about fifteen degrees centigrade," are PF-equivalent. The importance of the concepts of protoform and PF-equivalence derives in large measure from the fact that they serve as a basis for knowledge compression.

A knowledge base is assumed to consist of a factual database, FDB, and a deduction database, DDB. In both FDB and DDB, knowledge is assumed to fall into two categories: (a) crisp and (b) fuzzy. Examples of crisp items of knowledge in FDB might be: "Height of the Eiffel tower is 324m" and "Paris is the capital of France." Examples of fuzzy items might be "Most Swedes are tall," and "California has a temperate climate." Similarly, in DDB, an example of a crisp rule might be "If A and B are crisp convex sets, then their intersection is a crisp convex set." An example of a fuzzy rule might be "If A and B are fuzzy convex sets, then their intersection is a fuzzy convex set." A fuzzy rule may be a crisp assertion about fuzzy sets or a fuzzy assertion about crisp sets or a fuzzy assertion about fuzzy sets.

The deduction database is assumed to consist of a logical database and a computational database, with the rules of deduction having the structure of protoforms. An example of a computational rule is "If Q_1 A's are B's and Q_2 (A and B)'s are C's," then " $Q_1 Q_2$ A's are (B and C)'s," where Q_1 and Q_2 are fuzzy quantifiers, and A, B and C are labels of fuzzy sets. The number of rules in the computational database is assumed to be very large in order to allow a chaining of rules that may be query-relevant.

A very simple example of deduction in the prototype-centered approach—an example which involves string matching but no chaining -- is the following. Suppose that a query is "How many Swedes are very tall?" A protoform of this query is: $?Q$ A's are 2B , where Q is a fuzzy quantifier and 2B is assumed to represent the meaning of "very B," with the membership function of 2B being the square of the membership function of B. Searching DDB, we find the rule "If Q A's are B then $Q^{1/2}$ A's are 2B ," whose consequent matches the query, with $?Q$ instantiated to $Q^{1/2}$, A to "Swedes" and B to "tall." Furthermore, in FDB, we find the fact "Most Swedes are tall," which matches the antecedent of the rule, with Q instantiated to "Most." A to "Swedes" and B to "tall." Consequently, the answer to the query is " $Most^{1/2}$ Swedes are very tall," where the membership function of " $Most^{1/2}$ " is the square root of Most in fuzzy arithmetic.

The rules of deduction in the prototype-centered approach involve generalized constraint propagation, with generalized constraints represented as protoforms. Deduction is carried out through backtracking from query to query-relevant information. The concept of protoform has some links to the concept of ontology in ontological engineering, but unlike the latter is rooted in fuzzy logic and perception-based probability theory.

What should be underscored is that the problem of adding deduction capability to search engines is many-faceted and complex. It would be unrealistic to expect rapid progress toward its solution.

Compromise Approach to Neuro – Fuzzy Systems

Leszek RUTKOWSKI, Krzysztof CPAŁKA

Technical University of Czestochowa, Department of Computer Engineering, Poland

Abstract. In the paper we derived new neuro – fuzzy structures. They are characterized by more flexibility in the design process of such systems. Based on the input – output data we learn not only parameters of membership functions but also a type of the systems and aggregating parameters. Our approach introduces more flexibility to the structure and learning of neuro – fuzzy systems.

Keywords: *fuzzy systems, neuro – fuzzy systems, Mamdani approach, logical approach*

1. Introduction

It is well known that fuzzy – logic control does not require a conventional model of the process in contradiction to the classical control techniques which are based on analytical or experimental models. Moreover, traditional controllers cannot incorporate the linguistic fuzzy information - coming from human experts - into their design. On the other hand fuzzy controllers suffer from the lack of learning properties. Therefore, in the literature several neuro – fuzzy systems have been developed (see e.g. [1, 2, 4 – 13, 15]). They exhibit advantages of neural networks and fuzzy systems. In particular, fuzzy – neural systems combine learning abilities of neural networks and natural language description of fuzzy systems. The structure of such systems depends on fuzzy implications: Mamdani – type implication

$$I_1(a,b) = \min\{a,b\} \quad (1)$$

or logical - type implication, e.g. binary implication

$$I_2(a,b) = \max\{1-a,b\} \quad (2)$$

Consequently, the existing neuro – fuzzy systems employ two different approaches and in the literature there are no suggestions which of them is superior. Therefore in this paper we combine both approaches and present a compromise approach to neuro – fuzzy systems design. Moreover, we incorporate a concept of S - OWA - OR and S - OWA - AND aggregating operators to design neuro – fuzzy systems. Our contribution follows:

1.1 Compromise fuzzy implication

We propose a compromise fuzzy implication given by

$$I(a,b) = (1-\nu)I_1(a,b) + \nu I_2(a,b) \quad (3)$$

where $\nu \in [0,1]$, and based on implication (3) we derive a compromise neuro – fuzzy system. It includes Mamdani – type, logical – type, more Mamdani – type than logical – type and more logical - type than Mamdani – type fuzzy inference systems. It should be emphasized that our neuro - fuzzy system can be optimised with respect to the parameters of fuzzy membership functions in the process of learning. Moreover, also the parameter ν , determining a type of the system, can be found in the process of learning. To our best

knowledge such result has not been presented in the literature yet. The compromise neuro – fuzzy systems developed in the paper are simulated on iris plants classification problem and diagnosis of breast cancer. In the sequel S, T and N denote S-norm, T-norm and negation, respectively. By making use of this notation formula (3) can be generalized to the form

$$I(a, b) = N(\nu)T\{a, b\} + \nu S\{N(a), b\} \quad (4)$$

where $N(\nu) = 1 - \nu$.

1.2 Soft compromise fuzzy implication

We combine formula (3) or (4) with S - OWA - OR and S - OWA – AND aggregating operators [15] in order to soften the following fuzzy operators in neuro – fuzzy systems design:

- the logical AND connective that is used to connect the antecedents in the individual rules,
- fuzzy implication operators,
- the aggregation procedures.

2. Used methods and approaches

In this paper, we consider multi - input, single - output fuzzy systems mapping $X \rightarrow Y$, where $X \subset \mathbb{R}^n$ and $Y \subset \mathbb{R}$.

The fuzzifier performs a mapping from the observed crisp input space $X \subset \mathbb{R}^n$ to the fuzzy sets defined in Y . The most commonly used fuzzifier is the singleton fuzzifier which maps $\bar{x} = [\bar{x}_1, \dots, \bar{x}_n] \in X$ into a fuzzy set $A' \subset X$ characterized by the membership function

$$\mu_{A'}(x) = \begin{cases} 1 & \text{if } x = \bar{x} \\ 0 & \text{if } x \neq \bar{x} \end{cases} \quad (5)$$

The fuzzy rule base consists of a collection of N fuzzy IF - THEN rules in the form

$$R^{(k)}: \text{IF } x_1 \text{ is } A_1^k \text{ AND } x_2 \text{ is } A_2^k \text{ AND } \dots \text{AND } x_n \text{ is } A_n^k \text{ THEN } y \text{ is } B^k \quad (6)$$

or

$$R^{(k)}: \text{IF } x \text{ is } A^k \text{ THEN } y \text{ is } B^k \quad (7)$$

where $x = [x_1, \dots, x_n] \in X$, $y \in Y$, $A_1^k, A_2^k, \dots, A_n^k$ are fuzzy sets characterized by membership functions $\mu_{A_i^k}(x_i)$, whereas B^k are fuzzy sets characterized by membership functions $\mu_{B^k}(y)$, respectively, $k = 1, \dots, N$.

The fuzzy inference determines a mapping from the fuzzy sets in the input space X to the fuzzy sets in the output space Y . Each of N rules (7) determines a fuzzy set $\bar{B}^k \subset Y$ given by the compositional rule of inference

$$\bar{B}^k = A' \circ (A^k \rightarrow B^k) \quad (8)$$

where $A' = A_1^k \times A_2^k \times \dots \times A_n^k$. Fuzzy sets \bar{B}^k , according to the formula (8), are characterized by membership functions expressed by the sup-star composition

$$\mu_{\bar{B}^k}(y) = \sup_{x \in X} \{ \mu_{A'}(x) * \mu_{A_1^k \times \dots \times A_n^k \rightarrow B^k}(x, y) \} \quad (9)$$

where $*$ can be any operator in the class of T-norms. It is easily seen that for a crisp input $\bar{x} \in X$, i.e. a singleton fuzzifier (5), formula (9) becomes

$$\mu_{\bar{B}^k}(y) = \mu_{A_1^k \times \dots \times A_n^k \rightarrow B^k}(\bar{x}, y) = \mu_{A^k \rightarrow B^k}(\bar{x}, y) = I(\mu_{A^k}(\bar{x}), \mu_{B^k}(y)) \quad (10)$$

where $I(\cdot)$ is a fuzzy implication.

The aggregation operator, applied in order to obtain the fuzzy set B' based on fuzzy sets \bar{B}^* , is the T-norm or S-norm operator, depending on the type of fuzzy implication.

The defuzzifier performs a mapping from a fuzzy set B' to a crisp point \bar{y} in $Y \subset \mathbf{R}$. The COA (centre of area) method is defined by the following formula

$$\bar{y} = \frac{\int_Y y \mu_{B'}(y) dy}{\int_Y \mu_{B'}(y) dy} \quad (11)$$

or by

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}^r \cdot \mu_{B'}(\bar{y}^r)}{\sum_{r=1}^N \mu_{B'}(\bar{y}^r)} \quad (12)$$

in the discrete form, where \bar{y}^r denotes centres of the membership functions $\mu_{B'}(y)$, i.e. for $r = 1, \dots, N$

$$\mu_{B'}(\bar{y}^r) = \max_{y \in Y} \{\mu_{B'}(y)\} \quad (13)$$

2.1 Compromise approach

In this paper the fuzzy implication (10) is a T-norm (e.g. minimum or product) or S-implication (e.g. binary implication, known as the Kleene – Dienes implication) in the form

$$I(\mu_{A^*}(\bar{x}), \mu_{B^*}(\bar{y}^r)) = \begin{cases} T\{\mu_{A^*}(\bar{x}), \mu_{B^*}(\bar{y}^r)\} = \min\{\mu_{A^*}(\bar{x}), \mu_{B^*}(\bar{y}^r)\} & \text{for Mamdani approach} \\ S\{N(\mu_{A^*}(\bar{x})), \mu_{B^*}(\bar{y}^r)\} = \max\{1 - \mu_{A^*}(\bar{x}), \mu_{B^*}(\bar{y}^r)\} & \text{for logical approach} \end{cases} \quad (14)$$

and the aggregated output fuzzy set $B' \subset Y$ is given by

$$\mu_{B'}(\bar{y}^r) = \begin{cases} S\left\{T\{\mu_{A^*}(\bar{x}), \mu_{B^*}(\bar{y}^r)\}\right\} & \text{for Mamdani approach} \\ T\left\{S\{N(\mu_{A^*}(\bar{x})), \mu_{B^*}(\bar{y}^r)\}\right\} & \text{for logical approach} \end{cases} \quad (15)$$

Consequently formula (12) takes the form

$$\bar{y} = \begin{cases} \frac{\sum_{r=1}^N \bar{y}^r \cdot S\left\{T\left\{\prod_{i=1}^n \mu_{A_i^*}(\bar{x}_i), \mu_{B^*}(\bar{y}^r)\right\}\right\}}{\sum_{r=1}^N S\left\{T\left\{\prod_{i=1}^n \mu_{A_i^*}(\bar{x}_i), \mu_{B^*}(\bar{y}^r)\right\}\right\}} & \text{for Mamdani approach} \\ \frac{\sum_{r=1}^N \bar{y}^r \cdot T\left\{S\left\{N\left\{\prod_{i=1}^n \mu_{A_i^*}(\bar{x}_i)\right\}, \mu_{B^*}(\bar{y}^r)\right\}\right\}}{\sum_{r=1}^N T\left\{S\left\{N\left\{\prod_{i=1}^n \mu_{A_i^*}(\bar{x}_i)\right\}, \mu_{B^*}(\bar{y}^r)\right\}\right\}} & \text{for logical approach} \end{cases} \quad (16)$$

The compromise neuro – fuzzy system is based on formulas (14) – (16) combining with (4). We define this system as follows:

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}' \cdot \text{agr}_r(\bar{x}, \bar{y}')}{\sum_{r=1}^N \text{agr}_r(\bar{x}, \bar{y}')} \quad (17)$$

$$\text{agr}_r(\bar{x}, \bar{y}') = \mu_{B_r}(\bar{y}') = \left(N(\nu) S\{I_{1,r}(\bar{x}, \bar{y}'), \dots, I_{N,r}(\bar{x}, \bar{y}')\} + \right. \\ \left. + \nu T\{I_{1,r}(\bar{x}, \bar{y}'), \dots, I_{N,r}(\bar{x}, \bar{y}')\} \right) \quad (18)$$

$$I_{k,r}(\bar{x}, \bar{y}') = I(\mu_{A_k}(\bar{x}), \mu_{B_r}(\bar{y}')) = \left(N(\nu) T\{\tau_k(\bar{x}), \mu_{B_r}(\bar{y}')\} + \right. \\ \left. + \nu S\{N(\tau_k(\bar{x})), \mu_{B_r}(\bar{y}')\} \right) \quad (19)$$

$$\tau_k(\bar{x}) = \mu_{A_k}(\bar{x}) = T\{\mu_{A_k}(\bar{x}_1), \dots, \mu_{A_k}(\bar{x}_n)\} \quad (20)$$

where $\nu \in [0,1]$ and $N(a) = 1 - a$.

The general architecture of the system is depicted in Fig. 1. Note that this architecture has a multilayer structure and can be trained by the back – propagation method.

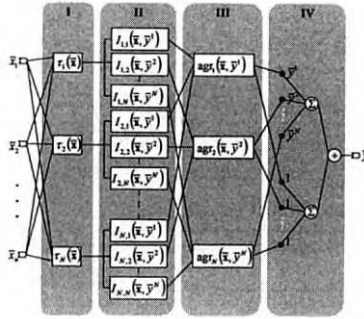


Figure 1. General architecture of neuro – fuzzy system

2.2 Soft compromise approach

In soft compromise approach we use formulas (18) – (20) with S - OWA - OR and S - OWA - AND aggregating operators [15] and define our system as follows:

$$\bar{y} = \frac{\sum_{r=1}^N \bar{y}' \cdot \text{agr}_r(\bar{x}, \bar{y}')}{\sum_{r=1}^N \text{agr}_r(\bar{x}, \bar{y}')} \quad (21)$$

$$\text{agr}_r(\bar{x}, \bar{y}') = \left(N(\alpha^{\text{or}}) \frac{1}{N} \sum_{k=1}^N I_{k,r}(\bar{x}, \bar{y}') + \right. \\ \left. + \alpha^{\text{or}} \left(N(\nu) S\{I_{1,r}(\bar{x}, \bar{y}'), \dots, I_{N,r}(\bar{x}, \bar{y}')\} + \right. \right. \\ \left. \left. + \nu T\{I_{1,r}(\bar{x}, \bar{y}'), \dots, I_{N,r}(\bar{x}, \bar{y}')\} \right) \right) \quad (22)$$

$$I_{k,r}(\bar{x}, \bar{y}') = \left(N(\alpha') \frac{1}{2} (\tilde{N}_{1-\nu}(\tau_k(\bar{x})) + \mu_{B_r}(\bar{y}')) + \right. \\ \left. + \alpha' \left(N(\nu) T\{\tau_k(\bar{x}), \mu_{B_r}(\bar{y}')\} + \right. \right. \\ \left. \left. + \nu S\{N(\tau_k(\bar{x})), \mu_{B_r}(\bar{y}')\} \right) \right) \quad (23)$$

$$\tau_k(\bar{x}) = \left(\begin{array}{l} N(\alpha^r) \frac{1}{n} \sum_{i=1}^n \mu_{A_i}(\bar{x}_i) + \\ + \alpha^r T\{\mu_{A_1}(\bar{x}_1), \dots, \mu_{A_n}(\bar{x}_n)\} \end{array} \right) \quad (24)$$

where $v \in [0,1]$, $\alpha^r, \alpha^l, \alpha^{agr} \in [0,1]$, $\tilde{N}_v(a) = N(v)N(a) + va$ and $N(a) = \tilde{N}_0(a) = 1 - a$.

3. Experiments and their results

Soft compromise neuro – fuzzy systems described by formulas (21) – (24) are simulated on iris plants classification problem and diagnosis of breast cancer.

3.1 Iris plants classification problem

Iris plants database, collected by Fisher [3], embraces 150 elements and each of them contains the following attributes: sepal length in cm, sepal width in cm, petal length in cm, petal width in cm, class (Iris Setosa, Iris Versicolour, Iris Virginica). In our experiments all sets are divided into a learning sequence (105 sets) and testing sequence (45 sets). We used the soft compromise neuro – fuzzy system with probabilistic triangular norms, Gaussian membership functions and two rules. The initial value of parameter v was set to 0.5.

In the learning process the parameter v reached the value equal 1 and the system became a logical type. The performance of the classification system was 97.78%.

3.2 Diagnosis of breast cancer

The breast cancer database was obtained from the University of Wisconsin Hospitals, Madison and presented by Dr. William H. Wolberg [14]. This database includes 699 elements (we removed 16 elements which have a single missing attribute) each of them contains the following attributes: Clump Thickness (1 – 10), Uniformity of Cell Size (1 – 10), Uniformity of Cell Shape (1 – 10), Marginal Adhesion (1 – 10), Single Epithelial Cell Size (1 – 10), Bare Nuclei (1 – 10), Bland Chromatin (1 – 10), Normal Nucleoli (1 – 10), Mitoses (1 – 10), Class (benign, malignant). In our experiments all sets are divided into a learning sequence (478 sets) and testing sequence (205 sets). We used the soft compromise neuro – fuzzy system with probabilistic triangular norms, Gaussian membership functions and two rules. The initial value of parameter v was set to 0.5.

In the learning process the parameter v reached the value equal 0 and the system became a Mamdani type. The performance of the classification system was 96.64%.

	Initial values	After learning	Mistakes (learning sequence)	Mistakes (testing sequence)
v	0.50	1.00		
α^r	0.75	0.00	1	1
α^l	0.75	1.00	(0.95%)	(2.22%)
α^{agr}	0.75	0.85		
v	0.00 (without learning)		2 (1.90%)	3 (6.66%)

Table 1. Iris plants classification problem

	Initial values	After learning	Mistakes (learning sequence)	Mistakes (testing sequence)
v	0.50	0.00		
α^r	0.75	0.95	10	7
α^l	0.75	0.94	(2.19%)	(3.36%)
α^{sep}	0.75	0.98		
v	1.00		14	2
	(without learning)		(2.92%)	(1.90%)

Table 2. Diagnosis of breast cancer

4. Conclusion

In the paper we derived soft compromise neuro – fuzzy systems based on Mamdani or logical type of fuzzy implications. The results can be easily extended to the case of Takagi – Sugeno systems or relational fuzzy systems.

References

- [1] K. Cpałka, *Flexible Neuro - Fuzzy Systems*, Ph. D. dissertation, Czestochowa 2001.
- [2] K. Rutkowski, L. Rutkowski, *Soft neuro – fuzzy systems*, Proceedings of the Fifth Conference On Neural Networks and Soft Computing, Zakopane 2000, pp. 296 - 301.
- [3] R. A. Fisher, *The use of multiple measurements in taxonomic problems*, Annu. Eugenics, vol. 7, 1936, pp. 179 – 188.
- [4] J. S. R. Jang, C. T. Sun, E. Mizutani, *Neuro – Fuzzy and Soft Computing*, Prentice Hall, Englewood Cliffs, 1997.
- [5] N. K. Kasobov, *Foundations of Neural Networks, Fuzzy Systems and Knowledge Engineering*, The MIT Press, Cambridge, MA, 1996.
- [6] E. H. Mamdani, S. Assilian, *An experiment in linguistic synthesis with a fuzzy logic controller*, International Journal of Man – Machine Studies, 1975, Vol. 7(1), pp. 1 – 13.
- [7] D. Nauck, F. Klawonn, R. Kruse, *Foundations of Neuro – Fuzzy Systems*, John Wiley & Sons, 1997.
- [8] D. Rutkowska, *Neuro – Fuzzy Architectures and Hybrid Learning*, Springer-Verlag 2001.
- [9] D. Rutkowska, R. Nowicki, *Implication – based neuro – fuzzy architectures*, Int. J. Appl. Math. Comput. Sci., 2000, Vol. 10, No. 4, pp. 675-701.
- [10] D. Rutkowska, M. Piliński, M. Rutkowski., *Neural Networks, Genetic Algorithms and Fuzzy Systems*, PWN, Warszawa (in Polish), 1997.
- [11] L. Rutkowski, K. Cpałka, *A general approach to neuro - fuzzy systems*, Proceedings of the 10th IEEE International Conference on Fuzzy Systems, Melbourne 2001.
- [12] L. Rutkowski, K. Cpałka, *Flexible structures of neuro – fuzzy systems*, Quo Vadis Computational Intelligence, Studies in Fuzziness and Soft Computing, Vol. 54, Springer 2000, pp. 479 – 484.
- [13] L. X. Wang, *Adaptive Fuzzy Systems and Control*, Prentice Hall, Englewood Cliffs, 1994.
- [14] W. H. Wolberg, *Wisconsin prognostic breast cancer*, <http://www.ics.uci.edu/pub/machine-learning-databases>.
- [15] R. R. Yager, D. P. Filev, *Essentials of Fuzzy Modelling and Control*, John Wiley & Sons, New York 1994.

On uninorms and their residual implicators

János Fodor, Bernard De Baets

Dept. of Biomathematics and Informatics, Faculty of Veterinary Sci.

Szent István University,

István u. 2, H-1078 Budapest, Hungary

E-mail: jfodor@univet.hu

Dept. of Applied Mathematics, Biometrics and Process Control,

Ghent University,

Coupure links 653, B-9000 Gent, Belgium

E-mail: Bernard.DeBaets@rug.ac.be

Abstract. Uninorms are an important generalization of t-norms and t-conorms, having a neutral element lying anywhere in the unit interval. A uninorm shows a typical block structure and is built from a t-norm, a t-conorm and a compensatory operator. Two important classes of uninorms are characterized, corresponding to the use of the minimum operator (the class U_{\min}) and maximum operator (the class U_{\max}) as mean operator. The characterization of representable uninorms, i.e. uninorms with an additive generator, is recalled. A residual operator is associated with a uninorm and it is characterized when it yields an impicator. The block structure of the residual impicator of members of the class U_{\min} is investigated. An explicit expression for the residual impicator of a representable uninorm is given, and important additional properties are listed.

Keywords: *t-conorm, t-norm, uninorm, representable uninorm, additive generator, residual impicator*

1 Uninorms and their structure

Uninorms have been introduced by Yager and Rybalov as a generalization of both t-norms and t-conorms [13]. Uninorms allow for a neutral element lying anywhere in the unit interval rather than at one or zero as is the case of t-norms and t-conorms [12]. Throughout this paper, a $[0, 1]^2 \rightarrow [0, 1]$ mapping is called a binary operator.

Definition 1 ([13]). A uninorm \mathcal{U} is an increasing, associative and commutative binary operator that satisfies

$$(\exists e \in [0, 1])(\forall x \in [0, 1])(\mathcal{U}(x, e) = x).$$

The element e corresponding to a uninorm \mathcal{U} is clearly unique and is called the neutral element of \mathcal{U} . Obviously, the case $e = 1$ leads back to t-norms, while the case $e = 0$ leads back to t-conorms. The structure of uninorms has been studied by Fodor *et al.* [10].

Proposition 1 ([10]). Consider a uninorm \mathcal{U} with neutral element $e \in]0, 1[$, then:

(i) the binary operator \mathcal{T}_U defined by

$$\mathcal{T}_U(x, y) = \frac{\mathcal{U}(ex, ey)}{e}$$

is a *t-norm*;

(ii) the binary operator \mathcal{S}_U defined by

$$\mathcal{S}_U(x, y) = \frac{\mathcal{U}(e + (1 - e)x, e + (1 - e)y) - e}{1 - e}$$

is a *t-conorm*.

The structure of a uninorm with neutral element e on the squares $[0, e]^2$ and $[e, 1]^2$ is therefore closely related to *t-norms* and *t-conorms*. For $e \in]0, 1[$, we denote by ϕ_e and ψ_e the linear transformations defined by $\phi_e(x) = \frac{x}{e}$ and $\psi_e(x) = \frac{x-e}{1-e}$. According to Proposition 1, to any uninorm \mathcal{U} with neutral element $e \in]0, 1[$, there corresponds a *t-norm* \mathcal{T} and a *t-conorm* \mathcal{S} such that:

- (i) $(\forall (x, y) \in [0, e]^2)(\mathcal{U}(x, y) = \phi_e^{-1}(\mathcal{T}(\phi_e(x), \phi_e(y))))$;
- (ii) $(\forall (x, y) \in [e, 1]^2)(\mathcal{U}(x, y) = \psi_e^{-1}(\mathcal{S}(\psi_e(x), \psi_e(y))))$.

Concerning the other parts of the unit square, we have the following proposition.

Proposition 2 ([13]). Consider a uninorm \mathcal{U} with neutral element e , then

$$(\forall (x, y) \in [0, e] \times [e, 1] \cup [e, 1] \times [0, e])(\min(x, y) \leq \mathcal{U}(x, y) \leq \max(x, y)).$$

Proposition 3 ([10]). For any uninorm \mathcal{U} , one of the following two cases always holds:

- (i) \mathcal{U} is a conjunctive (or and-like) uninorm: $\mathcal{U}(0, 1) = \mathcal{U}(1, 0) = 0$;
- (ii) \mathcal{U} is a disjunctive (or or-like) uninorm: $\mathcal{U}(0, 1) = \mathcal{U}(1, 0) = 1$.

2 Important classes of uninorms

2.1 The classes U_{\min} and U_{\max}

According to Proposition 3, a uninorm \mathcal{U} with neutral element e acts on $[0, e] \times [e, 1]$ and $[e, 1] \times [0, e]$ as a *compensatory operator*. The following two propositions investigate the extreme cases where it acts as the minimum operator or as the maximum operator on $[0, e] \times [e, 1]$ $[0, e]$.

Theorem 1. A binary operator \mathcal{U} is a conjunctive uninorm with neutral element $e \in]0, 1[$ such that $\mathcal{U}(\cdot, 1)$ is continuous on $[0, e]$ if and only if there exists a *t-norm* \mathcal{T} and a *t-conorm* \mathcal{S} such that

$$\mathcal{U}(x, y) = \begin{cases} \phi_e^{-1}(\mathcal{T}(\phi_e(x), \phi_e(y))) & , \text{ if } (x, y) \in [0, e]^2 \\ \psi_e^{-1}(\mathcal{S}(\psi_e(x), \psi_e(y))) & , \text{ if } (x, y) \in [e, 1]^2 \\ \min(x, y) & , \text{ elsewhere} \end{cases}$$

For obvious reasons, the class of uninorms characterized by Theorem 1 is denoted U_{\min} . A similar class of disjunctive uninorms, denoted U_{\max} , is characterized in the next theorem. The implications from left to right in Theorems 1 and 2 were shown in [10]; the implications from right to left are a matter of direct verification.

Theorem 2. *A binary operator \mathcal{U} is a disjunctive uninorm with neutral element $e \in]0, 1[$ such that $\mathcal{U}(\cdot, 0)$ is continuous on $]e, 1]$ if and only if there exists a t-norm \mathcal{T} and a t-conorm \mathcal{S} such that*

$$\mathcal{U}(x, y) = \begin{cases} \phi_e^{-1}(\mathcal{T}(\phi_e(x), \phi_e(y))) & , \text{ if } (x, y) \in [0, e]^2 \\ \psi_e^{-1}(\mathcal{S}(\psi_e(x), \psi_e(y))) & , \text{ if } (x, y) \in [e, 1]^2 \\ \max(x, y) & , \text{ elsewhere} \end{cases}$$

Given a t-norm \mathcal{T} and t-conorm \mathcal{S} , Theorems 1 and 2 show how to construct a conjunctive and disjunctive uninorm that have \mathcal{T} and \mathcal{S} as underlying t-norm and t-conorm. The class of all uninorms that have continuous \mathcal{T} and \mathcal{S} as underlying t-norm and t-conorm has been characterized recently (see [9]). In case of a strict t-norm and strict t-conorm, yet another construction exists, as will be shown in the next subsection.

Proposition 4 ([10]). *Consider a uninorm \mathcal{U} with neutral element e . It then holds that*

$$\underline{\mathcal{U}}_e \leq \mathcal{U} \leq \overline{\mathcal{U}}_e,$$

with $\underline{\mathcal{U}}_e \in U_{\min}$ and $\overline{\mathcal{U}}_e \in U_{\max}$ defined by

$$\begin{aligned} \underline{\mathcal{U}}_e(x, y) &= \begin{cases} 0 & , \text{ if } (x, y) \in [0, e]^2 \\ \max(x, y) & , \text{ if } (x, y) \in [e, 1]^2 \\ \min(x, y) & , \text{ elsewhere} \end{cases} \\ \overline{\mathcal{U}}_e(x, y) &= \begin{cases} \min(x, y) & , \text{ if } (x, y) \in [0, e]^2 \\ 1 & , \text{ if } (x, y) \in [e, 1]^2 \\ \max(x, y) & , \text{ elsewhere} \end{cases} \end{aligned}$$

2.2 Representable uninorms

In analogy to the representation of continuous Archimedean t-norms and t-conorms, Fodor et al. [10] have investigated the existence of uninorms with a similar representation in terms of a single variable function. This search turns out to lead back to the class of aggregative operators introduced by Dombi in [5]. This work is also closely related to that of Klement et al. on associative compensatory operators [11].

Proposition 5 ([10]). *Consider $e \in]0, 1[$ and a strictly increasing continuous $[0, 1] \rightarrow \overline{\mathbb{R}}$ mapping h with $h(0) = -\infty$, $h(e) = 0$ and $h(1) = +\infty$. The binary operator \mathcal{U} defined by*

$$\mathcal{U}(x, y) = h^{-1}(h(x) + h(y)), \quad \forall (x, y) \in [0, 1]^2 \setminus \{(0, 1), (1, 0)\},$$

and either $\mathcal{U}(0, 1) = \mathcal{U}(1, 0) = 0$ or $\mathcal{U}(1, 0) = \mathcal{U}(0, 1) = 1$, is a uninorm with neutral element e .

The mapping h in the foregoing proposition is called an additive generator of the uninorm \mathcal{U} .

Theorem 3 ([10]). *Consider a uninorm \mathcal{U} with neutral element $e \in]0, 1[$. There exists a strictly increasing continuous $[0, 1] \rightarrow \mathbb{R}$ mapping h with $h(0) = -\infty$, $h(e) = 0$ and $h(1) = +\infty$ such that*

$$\mathcal{U}(x, y) = h^{-1}(h(x) + h(y)), \quad \forall (x, y) \in [0, 1]^2 \setminus \{(0, 1), (1, 0)\}$$

if and only if

- (i) \mathcal{U} is strictly increasing and continuous on $]0, 1[^2$;
- (ii) there exists an involutive negator \mathcal{N} with fixed point e , i.e. $\mathcal{N}(e) = e$, such that

$$(\forall (x, y) \in [0, 1]^2 \setminus \{(0, 1), (1, 0)\}) (\mathcal{U}(x, y) = \mathcal{N}(\mathcal{U}(\mathcal{N}(x), \mathcal{N}(y)))).$$

The uninorms characterized by Theorem 3 are called representable uninorms. Note that any representable uninorm comes in a conjunctive and a disjunctive version, i.e. there always exist two representable uninorms that only differ in the points $(0, 1)$ and $(1, 0)$. The disjunctive version of a conjunctive representable uninorm \mathcal{U} will be denoted \mathcal{U}^* . Note that the partial mapping $\mathcal{U}(\cdot, 1)$ is not continuous in 0 and that the partial mapping $\mathcal{U}^*(\cdot, 0)$ is not continuous in 1. It is immediately clear that the classes U_{\min} and U_{\max} do not contain representable uninorms. Note that the additive generator of a representable uninorm is unique up to a positive multiplicative constant.

The involutive negator $\mathcal{N}_{\mathcal{U}}$ corresponding to a representable uninorm \mathcal{U} with additive generator h , as mentioned in Theorem 3, is given by $\mathcal{N}_{\mathcal{U}}(x) = h^{-1}(-h(x))$.

Any representable uninorm \mathcal{U} with neutral element e satisfies

$$(\forall x \in]0, e[) (\mathcal{U}(x, x) < x) \quad \text{and} \quad (\forall x \in]e, 1[) (\mathcal{U}(x, x) > x),$$

an obvious generalization of the Archimedean property of continuous t-norms and t-conorms. Note that Theorem 3 implies that the t-norm and t-conorm underlying a representable uninorm are both strict.

On the other hand, we will show next that representable uninorms are in some sense also generalizations of nilpotent t-norms and nilpotent t-conorms. We first recall the following characterizations. A continuous t-norm \mathcal{T} is nilpotent if and only if there exists an involutive negator \mathcal{N} such that

$$(\forall x \in]0, 1[) (\mathcal{T}(x, \mathcal{N}(x)) = 0).$$

Similarly, a continuous t-conorm \mathcal{S} is nilpotent if and only if there exists an involutive negator \mathcal{N} such that

$$(\forall x \in]0, 1[) (\mathcal{S}(x, \mathcal{N}(x)) = 1).$$

Proposition 6. *Consider a representable uninorm \mathcal{U} with neutral element e , then there exists an involutive negator \mathcal{N} such that*

$$(\forall x \in]0, 1[) (\mathcal{U}(x, \mathcal{N}(x)) = e).$$

In order to prove this proposition, it suffices to consider the involutive negator $\mathcal{N}_{\mathcal{U}}$.

3 Residual implicators of uninorms

The residuation technique is well-known in fuzzy set theory for constructing new logical operators from given ones [2, 3, 6, 7]. For instance, for any t-norm \mathcal{T} , the binary operator $\mathcal{I}_{\mathcal{T}}$ defined by

$$\mathcal{I}_{\mathcal{T}}(x, y) = \sup\{z \mid z \in [0, 1] \wedge \mathcal{T}(x, z) \leq y\}$$

is an implicator, i.e. is hybrid monotonous (decreasing first and increasing second partial mappings) and satisfies $\mathcal{I}_{\mathcal{T}}(1, 0) = 0$ and $\mathcal{I}_{\mathcal{T}}(1, 1) = \mathcal{I}_{\mathcal{T}}(0, 0) = 1$. For the minimum operator M for instance, the residual implicator \mathcal{I}_M , also known as the Gödel implicator, is given by

$$\mathcal{I}_M(x, y) = \begin{cases} 1 & , \text{ if } x \leq y \\ y & , \text{ elsewhere} \end{cases}$$

Similarly, for a t-conorm \mathcal{S} we can consider the binary operator $\mathcal{R}_{\mathcal{S}}$ defined by

$$\mathcal{R}_{\mathcal{S}}(x, y) = \sup\{z \mid z \in [0, 1] \wedge \mathcal{S}(x, z) \leq y\},$$

but apart from its obvious hybrid monotonicity, this operator has no particular logical interpretation.

We can now consider the construction of a residual operator $\mathcal{I}_{\mathcal{U}}$ from a uninorm \mathcal{U} . For a comprehensive study see [4], where all the proofs of the following results can also be found.

Proposition 7. *Consider a uninorm \mathcal{U} with neutral element $e \in]0, 1[$. The binary operator $\mathcal{I}_{\mathcal{U}}$ defined by*

$$\mathcal{I}_{\mathcal{U}}(x, y) = \sup\{z \mid z \in [0, 1] \wedge \mathcal{U}(x, z) \leq y\}$$

is an implicator if and only if

$$(\forall z \in]e, 1[)(\mathcal{U}(0, z) = 0).$$

Corollary 1. *For any conjunctive uninorm \mathcal{U} or representable uninorm \mathcal{U} (conjunctive or disjunctive), the residual operator $\mathcal{I}_{\mathcal{U}}$ is an implicator.*

Apart from representable uninorms, no other disjunctive uninorms \mathcal{U} are known for which the residual operator $\mathcal{I}_{\mathcal{U}}$ yields an implicator. Note that the residual operator $\mathcal{I}_{\mathcal{U}}$ of a uninorm \mathcal{U} with neutral element e satisfies

$$(\forall y \in [0, 1])(\mathcal{I}_{\mathcal{U}}(e, y) = y),$$

a generalization of the neutrality principle which holds for the residual implicator of a t-norm.

4 Residual implicators of members of the class U_{\min}

The first question that comes to mind when studying residual implicators of uninorms is whether they also show some kind of block structure (or ordinal sum structure). The following theorem investigates this structure for members of the class U_{\min} .

Theorem 4. *Consider $\mathcal{U} \in U_{\min}$ with neutral element $e \in]0, 1[$ and underlying t-norm \mathcal{T} and t-conorm \mathcal{S} . The residual implicator $\mathcal{I}_{\mathcal{U}}$ is given by*

$$\mathcal{I}_{\mathcal{U}}(x, y) = \begin{cases} \phi_e^{-1}(\mathcal{I}_{\mathcal{T}}(\phi_e(x), \phi_e(y))) & , \text{ if } (x, y) \in [0, e]^2 \wedge y < x \\ \psi_e^{-1}(\mathcal{R}_{\mathcal{S}}(\psi_e(x), \psi_e(y))) & , \text{ if } (x, y) \in [e, 1]^2 \\ \mathcal{I}_M(x, y) & , \text{ elsewhere} \end{cases}$$

One might have expected in Theorem 4 that $\mathcal{I}_{\mathcal{U}}(x, y) = \phi_e^{-1}(\mathcal{I}_{\mathcal{T}}(\phi_e(x), \phi_e(y)))$, for any $(x, y) \in [0, e]^2$. However, in case $x \leq y$, this would yield $\mathcal{I}_{\mathcal{U}}(x, y) = e$, whereas Theorem 4 yields $\mathcal{I}_{\mathcal{U}}(x, y) = 1$.

5 Residual implicators of representable uninorms

For a continuous Archimedean t-norm \mathcal{T} with additive generator f , the residual impicator $\mathcal{I}_{\mathcal{T}}$ is given by $\mathcal{I}_{\mathcal{T}}(x, y) = f^{(-1)}(f(y) - f(x))$ [3]. The following theorem brings a similar result for representable uninorms.

Theorem 5. *Consider a representable uninorm \mathcal{U} with additive generator h , then its residual impicator $\mathcal{I}_{\mathcal{U}}$ is given by*

$$\mathcal{I}_{\mathcal{U}}(x, y) = \begin{cases} h^{-1}(h(y) - h(x)) & , \text{ if } (x, y) \in [0, 1]^2 \setminus \{(0, 0), (1, 1)\} \\ 1 & , \text{ elsewhere} \end{cases}$$

Proposition 8. *Consider a representable uninorm \mathcal{U} , then $\mathcal{I}_{\mathcal{U}}$ is contrapositive w.r.t. $\mathcal{N}_{\mathcal{U}}$, i.e.*

$$(\forall (x, y) \in [0, 1]^2) (\mathcal{I}_{\mathcal{U}}(x, y) = \mathcal{I}_{\mathcal{U}}(\mathcal{N}_{\mathcal{U}}(y), \mathcal{N}_{\mathcal{U}}(x))).$$

Note that the involutive negator $\mathcal{N}_{\mathcal{U}}$ corresponding to a representable uninorm \mathcal{U} with neutral element e can be written as $\mathcal{N}_{\mathcal{U}}(x) = \mathcal{I}_{\mathcal{U}}(x, e)$.

For any disjunctive uninorm \mathcal{U} and negator \mathcal{N} , the binary operator $\mathcal{I}_{\mathcal{U}, \mathcal{N}}$ defined by

$$\mathcal{I}_{\mathcal{U}, \mathcal{N}}(x, y) = \mathcal{U}(\mathcal{N}(x), y)$$

clearly is an impicator.

Proposition 9. *Consider a conjunctive representable uninorm \mathcal{U} . Then the following equality holds:*

$$\mathcal{I}_{\mathcal{U}^*, \mathcal{N}_{\mathcal{U}}} = \mathcal{I}_{\mathcal{U}}.$$

Propositions 8 and 9 are generalizations of results that are characteristic for nilpotent t-norms [8]. Recall, for instance, that for a nilpotent t-norm \mathcal{T} , the residual impicator $\mathcal{I}_{\mathcal{T}}$ is contrapositive w.r.t. the involutive negator $\mathcal{N}_{\mathcal{T}}$ defined by $\mathcal{N}_{\mathcal{T}}(x) = \mathcal{I}_{\mathcal{T}}(x, 0)$. Moreover, it holds that $\mathcal{I}_{\mathcal{T}}(x, y) = \mathcal{S}(\mathcal{N}_{\mathcal{T}}(x), y)$, with \mathcal{S} the $\mathcal{N}_{\mathcal{T}}$ -dual t-conorm of \mathcal{T} .

Acknowledgment

This research has been supported in part by FKFP 0051/2000, and by the Bilateral Scientific and Technological Cooperation Flanders–Hungary BIL00/51 (B-08/2000).

References

- [1] B. De Baets, Model implicators and their characterization, in: N. Steele, Ed., *Proceedings of the First ICSC International Symposium on Fuzzy Logic (Zürich, Switzerland)* (ICSC Academic Press, 1995), pp. A42–A49.
- [2] B. De Baets, Residual operators of implicators, in: H.-J. Zimmermann, Ed., *Proceedings of the Third European Congress on Intelligent Techniques and Soft Computing (Aachen, Germany), Vol. 1*. (ELITE, 1995), pp. 136–140.

- [3] B. De Baets and R. Mesiar, Residual implicators of continuous t-norms, in: H.-J. Zimmermann, Ed., *Proceedings of the Fourth European Congress on Intelligent Techniques and Soft Computing (Aachen, Germany), Vol. 1.* (ELITE, 1996) pp. 27–31.
- [4] B. De Baets and J.C. Fodor, Residual operators of uninorms, *Soft Computing* **3** (1999), 89–100.
- [5] J. Dombi, Basic concepts for the theory of evaluation: The aggregative operator, *European Journal for Operations Research* **10** (1982) 282–293.
- [6] J. Fodor, On fuzzy implication operators, *Fuzzy Sets and Systems* **42** (1991) 293–300.
- [7] J. Fodor, A new look at fuzzy connectives, *Fuzzy Sets and Systems* **57** (1993) 141–148.
- [8] J. Fodor, Contrapositive symmetry of fuzzy implications, *Fuzzy Sets and Systems* **69** (1995)
- [9] J. Fodor, B. De Baets and T. Calvo, Characterization of uninorms with given underlying t-norms and t-conorms (2001) (submitted). 141–156.
- [10] J. Fodor, R. Yager and A. Rybalov, Structure of uninorms, *Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems* **5** (1997) 411–427.
- [11] E.-P. Klement, R. Mesiar and E. Pap, On the relationship of associative compensatory operators to triangular norms and conorms, *Int. J. Uncertainty, Fuzziness and Knowledge-Based Systems* **4** (1996) 129–144.
- [12] E.P. Klement, R. Mesiar and E. Pap, *Triangular Norms*, (Kluwer Academic Publishers, Boston/London/Dordrecht, 2000).
- [13] R. Yager and A. Rybalov, Uninorm aggregation operators, *Fuzzy Sets and Systems* **80** (1996) 111–120.

Adaptation of Fuzzy Controllers Based on Incremental Models

Ján Vaščák, Kaoru Hirota

Department of Cybernetics and Artificial Intelligence,

Faculty of Electrical Engineering and Informatics, Technical University of Košice,

Letná 9, 042 00 Košice, Slovakia, Jan.Vascak@tuke.sk

Interdisciplinary Graduate School of Science and Engineering,

Department of Computational Intelligence and Systems Science,

Tokyo Institute of Technology,

4259 Nagatsuta, Midori-ku, Yokohama 226-8502, Japan, hirota@hrt.dis.titech.ac.jp

Abstract. This paper deals with the description and analysis of the so-called Self-Organizing Fuzzy Logic Controller proposed by Procyk and Mamdani [5], which has been modified in many papers, e.g. [6] and [2] and belongs to main structures of adaptive fuzzy controllers. Its implementation is shown on LEGO robots where the 'parking problem' is solved to modify the concept of SOFLC with the intention to minimize its too excessive sensitivity to external signals.

Keywords: *Adaptive fuzzy controller, Self-organizing fuzzy controller, Incremental model, Jacobian, Performance*

1 Introduction

Fuzzy logic has found many successful applications, especially in the area of control, but there are some limits of its use that are connected with the inability of the knowledge acquisition and adaptation to changed external conditions or parameters of the controlled system. To overcome this problem there have been published lots of papers, e.g. [1], [4] and [5], which deal with structures of *Adaptive Fuzzy Controllers* (AFC) using mostly approaches based on many variations of gradient-descent methods, the least square method [3], linear and non-linear regression [7] or the linguistically based rule extraction.

In this paper we will describe and analyze the so-called *Self-Organizing Fuzzy Logic Controller* (SOFLC) proposed by Procyk and Mamdani [5], which has been modified in many papers, e.g. [6] and [2]. Further, its implementation will be shown on an example of LEGO robots. Results of experiments with these systems are summarized in the concluding part of this paper.

2 Structure of SOFLC

SOFLC (see fig. 1) belongs to the so-called *performance-adaptive controllers*, which evaluate the control quality by a criterion (or more criteria) like transition time, energy consumption, overshoots, etc. Such a quality measure is called a *performance measure* $p(k)$.

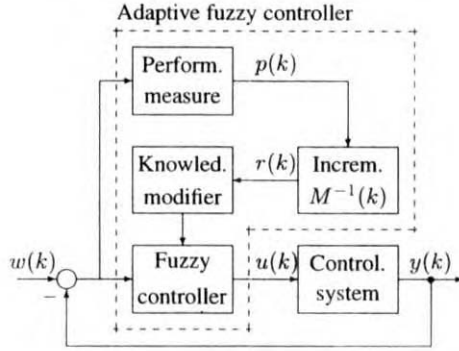


Figure 1: Self-organizing fuzzy logic controller.

Control criteria are contented in the block of *performance measure* where $p(k)$ expresses the magnitude and direction of changes to be performed in the knowledge base of the controller. The basic design problem of AFC consists in the design of M – a simplified *incremental model* of the controlled system $M = J.T$ (J – Jacobian, T – sampling period), which is computed in each time sample $t = K.T$ ($K = 0, 1, \dots$). It represents a supplement to the original model to reach a zero control error ($w(k) - y(k)$) and is analogous to the linear approximation of the first order differential equation or in other words, to gradients, too. As Jacobian is a determinant of all first derivatives of the system with n equations f_1, \dots, f_n of n input variables x_1, \dots, x_n it means J is equal to the determinant of the dynamics matrix, i.e. it is a numerical value describing all n gradients in the sense of a characteristic value.

Now we need to transform this incremental description of a controlled system to the description of a controlling system, i.e. a controller. Considering the properties of the feed back connection we can see that $y(k) \approx e(k)$ ($w(k)$ is known). As inputs and outputs of a controlled system change to outputs and inputs of a controller, respectively we can get the controller description like the inverse function of $y(k) = f_M(u(k))$, i.e. the model of the controller is $u(k) = f_M^{-1}(y(k))$. Because J is a crisp number then M^{-1} is the reverse value of $J.T$. The *reinforcement value* $r(k)$ is computed as $r(k) = M^{-1}.p(k)$ and represents the correction of the knowledge base.

Let the knowledge base in the time step k of such AFC be $R(k)$ and let its modification in next time step be $R(k+1)$. The general adaptation rule can be described like:

$$R(k+1) = (R(k) \cap \overline{R_{bad}(k)}) \cup R_{new}(k). \quad (1)$$

We see, firstly, the part of knowledge R_{bad} that caused the low quality control is removed from $R(k)$ and then it is completed by new knowledge R_{new} , which is corrected by $r(k)$. R_{bad} and R_{new} are computed as follows:

$$R_{bad}(k) = \text{fuzz}(x_1^*(k)) \times \dots \times \text{fuzz}(x_n^*(k)) \times \text{fuzz}(u^*(k)), \quad (2)$$

$$R_{new}(k) = \text{fuzz}(x_1^*(k)) \times \dots \times \text{fuzz}(x_n^*(k)) \times \text{fuzz}(u^*(k) + r(k)). \quad (3)$$

x_1, \dots, x_n are states of the controller, $u(k)$ is its output and $*$ denotes these values are crisp (to prevent possible misunderstanding). The only difference between R_{bad} and R_{new} is in the consequent part of IF – THEN rules, i.e. in the output, which confirms the role of $r(k)$ as a correction value. The implementation of the knowledge base adaptation can be either *rule-based* or *relation-based*. We chose the first kind of implementation. In the next section some properties of SOFLC will be described.

2.1 Advantages and Drawbacks of SOFLC

It seems to be reasonable to search for methods that would minimize the deviation from the optimal state as quickly as possible. Therefore gradient-based methods should be the most convenient for the knowledge modification. In such a sense SOFLC is also a special form embedding this calculus since it utilizes Jacobian. The only difference between SOFLC and 'classical' gradient-descent methods (GDA) can be described as follows. SOFLC represents *gradient of behavior* and 'classical' gradient methods can be related to the *gradient of the knowledge base*. In other words, SOFLC directly calculates the derivative of the system behavior, i.e. its change and 'classical' gradient methods compute the change of the control error in dependence on the knowledge base parameters. From this reason there is a close relation between *gradient of behavior* and *gradient of the knowledge base* but no equivalence, rather resemblance or similarity (\approx).

Although GDA should be the fastest adaptation two basic problems are related to it. Firstly, the error function $E(k)$ is unknown in advance and it may be of a complex shape with a number of local minima. It is very difficult to estimate their number and possible place of the global minimum, i.e. optimal solution in advance. Further, the absence of such an estimation disables the determination of the learning factor value, too. If it is too small, the convergence will be too slow, and if it is too big, there will be a risk the global minimum will be 'jumped over'. Secondly, there is a possibility to minimize only one criterion – error function $E(k)$ but in practice there are also other control criteria. SOFLC overcomes these problems partially and it is more practice-oriented, because it is able to involve other criteria, too. However, it is sensitive to external signals such as disturbances, noises and set-point changes [2] and [6] because it is not able to distinguish whether the parameters of the controlled system are changed or an external signal entered the system. A negative effect may occur, if the adaptation proceeds, although it is not (more) necessary. So, some wrong changes in the knowledge base may be performed. This state is caused by wrong understanding; if e.g. an external error occurs and AFC will evaluate it as a parameter change. In [6] it is shown that by adding some supervisory rules this problem can be solved. A modification of SOFLC in the form of the so-called sliding mode control is made in [2] where not only the position of the control error $e(t)$ but also its change (the first derivative) are taken into consideration. This method needs complete knowledge about the states of the controlled system and proper design of the sliding hyper-plane. However, how to design a 'good' hyper-plane is not solved in this approach. Further, the only criterion for the controller design is the control error. It is of course important that its value converges to zero but in many applications also another criteria may be still more important. From this reason we proposed a hybrid structure merging both SOFLC as well as GDA to balance their properties. This structure is described in the following section.

3 Hybrid Implementation of SOFLC for LEGO Robots

As already mentioned in section 2 knowledge base $R(k)$ can be described in two fundamental ways: *rule-based* and *relation-based*. In the first case there is a set of fuzzy IF – THEN rules and a set of definitions of linguistic terms in the form of membership functions. Let us denote such a set of rules in this section $R(k)$, too. $R(k)$ is the set of N_r fuzzy rules r_p ($p = 1, \dots, N_r$) of n inputs and one output. Such a rule r_p represents the Cartesian product of these input/output variables and is also a fuzzy relation $R_p = A_{1,p} x \dots x A_{n,p} x B_p$ where $A_{1,p}, \dots, A_{n,p}$ are linguistic values of x_1, \dots, x_n for the p -th rule. The knowledge base is then a union of such rules (fuzzy relations) and after substituting into (1) it will be changed for next step $k+1$ to (4). $R_{bad}(k)$ can be a union of all previously fired rules, too. However, for the sake of simplicity we will consider only one rule with the greatest strength α_p and therefore $A_1^{bad} x \dots x A_n^{bad}$ is its premise. The reinforcement value $r(k)$ corrects only the consequent of such a rule and B^{new} is the fuzzified result of $u(k) + r(k)$, i.e. $fuzz(u(k) + r(k))$. The simplest fuzzification is in the form of singletons but in general other forms are possible, too.

$$\begin{aligned}
 R(k+1) = & \left[\bigcup_{p=1}^{N_r} (A_{1,p} \cap \overline{A_1^{bad}}) x \dots x A_{n,p} x B_p \right] \cup \\
 & \dots \cup \left[\bigcup_{p=1}^{N_r} A_{1,p} x \dots x (A_{n,p} \cap \overline{A_n^{bad}}) x B_p \right] \cup \\
 & \cup \left[\bigcup_{p=1}^{N_r} A_{1,p} x \dots x A_{n,p} x (B_p \cap \overline{B^{bad}}) \right] \cup \\
 & \underbrace{\cup (A_1^{bad} x \dots x A_n^{bad} x B^{new})}_{R_{new}}
 \end{aligned} \tag{4}$$

In (4) following drawbacks can be seen:

1. *Possibility to change only one rule in one step* – The adaptation process will be longer and it is possible, that the low control quality was not caused by the rule with the greatest strength but by another rules working as noise. In every case the convergence will be worse.
2. *Growth of the rule number* – Let us consider 4 rules with two inputs and one output then there will be 13 rules in next step, which leads to an enormous growth of rules in the step $k+2, k+3$, etc. $(N_r(k+1) = N_r(k) \cdot (n+1) + 1)$
3. *Need of garbage collection* – Previous two points show us such filtering or 'garbage collection', i.e. removing useless rules is necessary to prevent the computational complexity and to improve the adaptation convergence.

To utilize the advantages of GDA and relation-based implementation of SOFLC we proposed special hybrid connection of these two methods where the control is steadily switched between SOFLC and gradient-descent adaptation. The adaptation process can be described in following steps:

1. Defining input and output variables.
2. Defining term sets for variables in the step 1.
3. Designing initial membership functions (not necessary).

4. Processing GDA until the prescribed threshold of the control error $e(k)$ is reached.
5. If $e(k)$ is under such a threshold then processing SOFLC (by a switch) otherwise jump (switch) to the step 4.

The main idea is that GDA is the fastest method if the threshold of the control error as the most important criterion is not too strict. In such a case we can choose a greater learning factor and speed up the adaptation. After this 'rough' adaptation we can switch the control to SOFLC to minimize the control error to be as small as possible and at this same time to include other criteria, too.

This hybrid control algorithm was implemented and tested on LEGO robots. Its results were compared also with a non-adaptive FC designed by a human operator. The control task was the so-called *parking problem*, i.e. to park a mobile robot at a given place and direction and was solved with and without obstacles. The process monitor evaluates the parking process by two criteria: *parking error* E_P – more important corresponding with the control error and *trajectory error* E_T – computed as division of the *real* trajectory length and *optimal* trajectory length. The optimal trajectory is the shortest distance between the robot and the goal. The first criterion is in the form:

$$E_P = \sqrt{(\phi_f - \phi)^2 + (x_f - x)^2 + (y_f - y)^2}. \quad (5)$$

(x, y) are coordinates of the robot ϕ is its position angle and (x_f, y_f, ϕ_f) are position and direction of the goal (parking place). Similar description is used for starting (initial) points, too. In the case of obstacles one additional criterion comes still into consideration – *number of impacts* on the obstacle.

The criterion predetermines also the structure of the IF – THEN rules, in our case three inputs (x, y, ϕ) and one output – change of the wheels angle for such a robot. The parking problem was solved with the help of a non-adaptive controller by 35 rules. It has no sense to observe the number of rules in the case of SOFLC because this number was very varying. We need to consider there is an unlimited number of combinations for intersections of fuzzy sets in (4). However, their number was considerably (several times) greater than for the non-adaptive controller. From this reason it was necessary to use a simple garbage collector, which minimizes the number of rules. It removes replaced and identical rules. If there are rules with identical premises but different consequences the older rule will be removed. Further, it will be possible to improve its efficiency if we remove such rules whose membership functions have little values of grade of membership or by merging rules with similar premises.

Results of several experiments for different starting points are depicted in figures 2 and 3. We can see the first two criteria E_P and E_T are better fulfilled at a non-adaptive FC. There are two reasons. Firstly, E_P and E_T are not mutually independent. Both are quantitative and E_P influences E_T directly proportionally. If E_P increases then also the trajectory will be more different from the optimal length but the shape may be in spite of that 'better' what is also this case. It can be seen especially at the obstacle avoidance (fig. 3). This assertion is supported by a smaller number of impacts at the adaptive FC than at the non-adaptive FC. Secondly, reinforced rules are fired in next steps after the error already occurred and in such a way delay influences the efficiency of such an adaptive FC negatively. This problem can be eliminated by shortening the sampling period T . There are only hardware limitations.

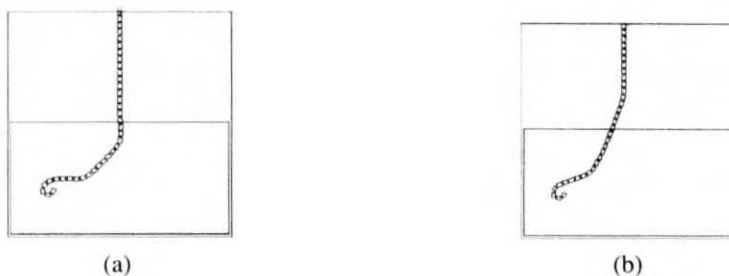


Figure 2: Comparison of trajectories for a non-adaptive FC (20, 80, 260) (a) and hybrid AFC (20, 80, 260) (b).



Figure 3: Comparison of trajectories with an obstacle for a non-adaptive FC (60, 70, 150) (a) and hybrid AFC (80, 80, 260) (b).

4 Conclusions

The principal advantage of both approaches is the substitution of a human expert in establishing the fundamental knowledge about the fuzzy controller, which is the most serious disadvantage of standard fuzzy systems. The designs presented enable fuzzy systems to be used for the control of systems showing great changes of parameters during their working. The second advantage is that we need to know only the dynamics of the controlled system and no input – output samples are necessary in advance. This fact enables the on-line adaptation and the set up of minimum parameters, which can lead to the decrease of the computational complexity. A hypothesis can be stated the rule-based implementation of SOFLC is more convenient for dynamical systems of higher order or with significant non-linearities but it has great demands on computational capacity.

References

- [1] Harris, C.J., Moore, C.G.: Intelligent identification and control for autonomous guided vehicles using adaptive fuzzy-based algorithms. *Engineering Applications of Artificial Intelligence* 2, 1989, pp. 267–285.
- [2] Kim, Y.T., Bien, Z.: Robust self-learning fuzzy controller design for a class of nonlinear MIMO systems. *Int. Journal Fuzzy Sets and Systems*, Elsevier Publisher, Holland, N. 2, Vol. 111, 2000, pp. 117–135.
- [3] Ma, M., Zhang, Y., Langholz, G., Kandel, A.: On direct construction of fuzzy systems. *Int. Journal Fuzzy Sets and Systems*, Elsevier Publisher, Holland, N. 1, Vol. 112, 2000, pp. 165–171.

- [4] Nauta Lemke, H.R., De-Zhao, W.: Fuzzy PID supervisor. IN: Proceedings of the 24-th IEEE Conference on Decision and Control, Fort Lauderdale, Florida, USA, 1985.
- [5] Procyk, T.J., Mamdani, E.H.: A linguistic self-organizing process controller. *Automatica* N. 15, 1979, pp. 15–30.
- [6] Zhang, B.S., Edmunds, J.M.: Selforganizing fuzzy logic controller. *IEEE Proc. D* 139 (5), pp. 460–464, 1992.
- [7] Várkonyi-Kóczy, A.R., Kovácsházy, T., Takács, O., Benedecsik, Cs.: Anytime Evaluation of Regression-Type Algorithms. *International Journal of Advanced Computational Intelligence*, Vol. 5, No. 1, pp. 2-7, Feb. 2001.

Optimal order of convergence for α -cut based fuzzy interpolators

Domonkos Tikk

Dept. of Telecommunications & Telematics,
Budapest University of Technology and Economics,
1117 Budapest, Magyar Tudósok Körútja 2., Hungary,
tikk@ttt.bme.hu

Intelligent Integrated Systems Japanese–Hungarian Laboratory,
1111 Budapest, Műegyetem rakpart 3., Hungary

Abstract. This paper investigates the approximation behaviour of the α -cut based fuzzy interpolators. First, it is shown that the so-called KH fuzzy interpolator is a fuzzy generalization of a well-known and thoroughly investigated parameterized interpolatory operator from approximation theory, the Shepard operator. Exploiting the aforementioned relationship, we establish analog results on the approximation rate of KH controllers. The optimal order and class of approximation (saturation problem) are determined for certain values of the parameter λ . Corresponding results on the MACI method, being a modification of the KH interpolator, are also provided.

Keywords: *Fuzzy inference systems, α -cut based fuzzy interpolators, KH interpolator, MACI method, Shepard operator, Universal approximation, Approximation error estimates, Saturation problem*

1 Introduction

Universal approximation theorems on soft computing techniques (see e.g. [3, 11, 21] for fuzzy systems, and [5, 6] for neural nets) have been usually criticized due to their solely existential nature [7, 20]. In the recent past, an effort has arisen to give constructive proofs, or to determine the number of “building blocks” (antecedents or rules in fuzzy, hidden neurons in neural terminology) as a function of the accuracy (see e.g. [2, 10, 12, 22]). Naturally, to obtain such results one has to restrict somehow the set of continuous functions, usually requiring some smoothness conditions on the approximated function. In approximation theory, the optimal order of convergence is called the saturation order, and the subset of continuous functions which can be approximated with the specified order is termed the saturation class. Hitherto, saturation classes and orders have not been determined for soft computing techniques.

In this paper we determine saturation order and classes for α -cut based fuzzy interpolators, namely for the stabilized KH interpolator [8, 19], and for stabilized MACI (Modified Alpha-Cut based Interpolation) interpolator [17, 18]. These fuzzy interpolators are the input-output functions of corresponding fuzzy rule based interpolation technique, having been proposed to provide an inference mechanism suitable for rule bases containing gaps.

2 Preliminaries: KH and Shepard interpolators

We note in advance that all results for KH interpolators can be carried over for MACI interpolator, using the fact that the latter tailors the conclusion as a finite sum of KH interpolators. Therefore we restrict our investigation mainly for KH interpolators.

The saturation problem of α -cut based interpolators is solved in two steps. First, it is proved that the stabilized KH interpolator is, in fact, the fuzzy generalization of the Shepard operator. The fuzziness of the conclusion is also determined in terms of the fuzziness of consequent sets. This statement is crucial in order to exploit the results obtained for Shepard interpolator in the case of KH interpolator.

Let us turn now to describe KH and Shepard interpolators. The original KH method was introduced in [8]. It creates the conclusion by means of its α -cuts based on the Extension Principle and the Resolution Principle. For every α value of the breakpoint level set, it determines the conclusion as

$$B_{\alpha C}^* = \frac{\sum_{i=1}^n B_{i\alpha C} \frac{1}{d_C(A_{\alpha C}^*, A_{i\alpha C})}}{\sum_{k=1}^n \frac{1}{d_C(A_{\alpha C}^*, A_{k\alpha C})}}, \quad (1)$$

where A_i and B_i ($i = 1, \dots, n$) denote the antecedents neighbouring the observation A^* , and the corresponding consequents fuzzy sets, respectively. $C \in \{L, U\}$, where L and U refer to the lower and upper extreme of the α -cut. Finally, the function $d_C : \mathbf{R} \times \mathbf{R} \rightarrow [0, +\infty)$ is an appropriate lower/upper distance function (cf. [9]). If $n = 2$, the approach is termed linear interpolation.

In [19], the stabilized version of the KH interpolation is introduced, when we take λ th power of the distance, where the exponent λ is not smaller than N , the dimension of the antecedents:

$$B_{\alpha C}^* = \frac{\sum_{i=1}^n B_{i\alpha C} \frac{1}{d_C^\lambda(A_{\alpha C}^*, A_{i\alpha C})}}{\sum_{k=1}^n \frac{1}{d_C^\lambda(A_{\alpha C}^*, A_{k\alpha C})}}, \quad (\lambda \geq N); \quad (2)$$

Now, we recall the universal approximation theorem of the stabilized KH approach [19].

Definition 1. Let $\mathbf{R}^N \supset \Omega = [a_1, b_1] \times \dots \times [a_N, b_N]$, further let $\{\Gamma_n\}_{n=1}^\infty$ be a sequence of finite subsets of Ω with $\#\Gamma_n = n$. If

$$\forall \varepsilon > 0 \exists n_0 \forall \omega \in \Omega \forall n \geq n_0 : \left| \frac{\#(\Gamma_n \cap \omega)}{\#\Gamma_n} - \frac{|\omega|}{|\Omega|} \right| < \varepsilon \quad (3)$$

then the set Γ_n are uniformly distributed on the domain Ω . Here $\#(\Gamma_n \cap \omega)$ denotes the cardinality of the finite set $(\Gamma_n \cap \omega)$ and $|\omega|$ is the Lebesgue measure of ω .

Theorem 2. Consider the L_p norm $\|\cdot\|_p$ with $p \in [1, \infty]$, the domain $\mathbf{R}^N \supset \Omega = [a_1, b_1] \times \dots \times [a_N, b_N]$ and a continuous function $f : \Omega \rightarrow \mathbf{R}$, then for all $x \in \Omega$ the expression

$$\lim_{n \rightarrow \infty} K_n^\lambda(f, x) := \lim_{n \rightarrow \infty} \sum_{k=1}^n f(x_k^{(n)}) \frac{\frac{1}{\|x - x_k^{(n)}\|_p^\lambda}}{\sum_{j=1}^n \frac{1}{\|x - x_j^{(n)}\|_p^\lambda}} \quad (4)$$

is equal to $f(x)$, where measurement points $x_k^{(n)}$ are uniformly distributed on Ω in the sense of (3), and $\lambda \geq N$.

In [19], the universal approximation property of $K_n^\lambda(f, x)$ was proven. From now, we mean the stabilized KH interpolator (2) on the term “KH interpolator”.

The Shepard interpolation method was first introduced in [13] for arbitrarily placed bi-variate data as

$$S_0(f, x, y) = \begin{cases} f(x_i, y_i), & \text{if } (x, y) = (x_i, y_i) \text{ for some } i \\ \left(\sum_{i=0}^n f(x_i, y_i) / d_i^\lambda \right) / \left(\sum_{i=0}^n 1 / d_i^\lambda \right), & \text{otherwise} \end{cases} \quad (5)$$

where measurement points x_i, y_i ($i = 0, \dots, n$) are irregularly spaced on the domain of $f \in \mathbf{R}^2 \rightarrow \mathbf{R}$, $\lambda > 0$, and $d_i = [(x - x_i)^2 + (y - y_i)^2]^{1/2}$. This function can be used typically when a surface model is required to interpolate scattered spatial measurements (e.g. pattern recognition, geology, cartography, earth sciences, fluid dynamics and many others).

Beside the application oriented investigation of Shepard's method such as [1], an increasing interest has arisen from mathematical researchers to examine the approximation property of formula (5). For a more general analysis, formula (5) was generalized to

$$S_n^\lambda(f, x) = \frac{\sum_{i=0}^n f(x_i)(x - x_i)^{-\lambda}}{\sum_{i=0}^n (x - x_i)^{-\lambda}}, \quad \lambda > 0, n = 1, 2, \dots \quad (6)$$

for an arbitrary $f \in C[0, 1]$, where x_i ($i = 0, \dots, n$), in general, denotes the nodes of the equidistant distribution of the domain $[0, 1]$. We recall that fixing the domain to the interval $[0, 1]$ does not mean any restriction.

The possible use of rational functions of type (6) as approximating means was first discovered by J. Balázs. (After his name this operator is often termed Balázs–Shepard operator in the literature of approximation theory.) The properties of the (6) operator was widely investigated; see e.g. [4, 14, 15, 16].

3 Main results

As it was shown in [19], for a fixed $\alpha \in [0, 1]$ and $C \in \{L, U\}$ the input-output function of the (stabilized) KH interpolator, $K_n^\lambda(f, x)$ coincides with the Shepard operator. Therefore, we can consider the family of $K_n^\lambda(f, x)$ functions as a generalization of the $S_n^\lambda(f, x)$.

Here, first we aim at clarifying what is nature of this generalization, how the approximated functions for various α and/or C values differ. It is obvious, that the family of KH functions tailor the same approximated function, if all the involved fuzzy sets are crisp. Next, we investigate how the conclusion depends on the fuzziness of the antecedents and consequents.

Proposition 3. *Under the conditions of Theorem 2, the fuzziness of the conclusion of the stabilized KH interpolator is bounded by the maximum fuzziness of the consequent fuzzy sets, if n , the number of knot points tends to ∞ .*

The analog result can be established for MACI interpolator. It is shown that the fuzziness of the conclusion determined by the MACI method is usually larger than the one created by the KH interpolator. In consequence, it is not surprising that the conclusion is more fuzzy even if the number of rules tends to infinity.

Proposition 4. Let k be the number of KH interpolators needed to create the MACI's conclusion, approximating the functions f_i with δ_i accuracy ($i = 1, \dots, k$). Then the conclusion's fuzziness is less than $\sum_{i=1}^k |\delta_i|$, if the number of knot points tends to ∞ .

Because of Propositions 3 and 4, the theorems for Shepard operators are convertible for the KH (or MACI) interpolators, bearing in mind that all derived results have an uncertainty factor (or fuzziness) due to the fuzziness of the consequents.

The approximation property and the saturation problem of the Shepard operator were investigated for various λ values in [4, 14, 15, 16]. Here we recall the theorem of Szabados [16], which gives a complete analysis for all $\lambda \geq 1$. When $\lambda < 1$, the operator does not converge for all $f(x) \in C[0, 1]$, so it is of no interest for our investigations.

Theorem 5. [16] The approximation order of the operator $S_n^\lambda(f, x)$ is

$$\|f(x) - S_n^\lambda(f, x)\| = \begin{cases} O(\omega(f, n^{-1})), & \text{if } \lambda > 2 \\ O(n^{1-\lambda} \int_{1/n}^1 t^{-\lambda} \omega(f, t) dt), & \text{if } 1 < \lambda \leq 2 \\ O(\log^{-1} n \int_{1/n}^1 t^{-\lambda} \omega(f, t) dt), & \text{if } \lambda = 1 \end{cases} \quad (7)$$

for any $f \in C[0, 1]$.

From Theorem 5 it is obvious that if $f(x) \in \text{Lip } 1$ and $\lambda > 2$, then the saturation order is

$$\|f(x) - S_n^\lambda(f, x)\| = O(n^{-1}). \quad (8)$$

In fact, as it is proved in [14], (8) holds if and only if $f(x) \in \text{Lip } 1$. Furthermore

$$\|f(x) - S_n^\lambda(f, x)\| = o(n^{-1}), \quad (9)$$

if and only if $f(x) = \text{const}$. Thus the saturation problem is completed for $\lambda > 2$.

If $\lambda = 2$, even with $f(x) \in \text{Lip } 1$ Theorem 5 gives only

$$\|f(x) - S_n^\lambda(f, x)\| = O\left(\frac{\log n}{n}\right). \quad (10)$$

This result can be improved to $O(n^{-1})$ under stronger restriction on $f(x)$:

Theorem 6. [16] If $f'(x) \in [0, 1]$ and

$$\int_0^1 t^{-1} \omega(f', t) dt < \infty, \quad (11)$$

further

$$f'(0) = f'(1) = 0 \quad (12)$$

then

$$\|f(x) - S_n^2(f, x)\| = O(n^{-1}) \quad (13)$$

Even less is known for the $1 \leq \lambda \leq 2$ case, when only best error estimates are obtained. From Theorem 5,

$$O(n^{1-\lambda}), \quad 1 < \lambda < 2 \quad \text{and} \quad O(\log^{-1} n), \quad \lambda = 1 \quad (14)$$

provided that

$$\int_0^1 t^{-\lambda} \omega(f, t) dt < \infty \quad (15)$$

holds.

The results of Theorems 5 and 6 can be carried over directly for KH and MACI interpolators under the following conditions:

1. The approximated function f is univariate, i.e. the input universe of the interpolator is one dimensional.
2. The knot points are uniformly distributed on the input space.

Corollary 7. *The KH interpolator (4) saturates with order $O(n^{-1})$ on the class of the functions $f(x) \in \text{Lip } 1$, if $\lambda > 2$, the knots points are uniformly distributed in the input space, and the input is one dimensional.*

Analog result holds for MACI interpolators.

Now, we can establish saturation order and class for KH interpolator.

Corollary 8. *The best approximation order of the KH interpolator $K_n^\lambda(f, x)$ ($1 \leq \lambda \leq 2$) is*

$$\|f(x) - K_n^\lambda(f, x)\| = \begin{cases} O(n^{-1}), & \text{if } \lambda = 2, \\ O(n^{1-\lambda}), & \text{if } 1 \leq \lambda < 2, \\ O(\log^{-1} n), & \text{if } \lambda = 1, \end{cases}$$

on the class of the functions $f(x) \in \text{Lip } 1$, if $f(x)$ further satisfies the conditions (11) and (12) for $\lambda = 2$, and the condition (15) for $1 \leq \lambda < 2$, and furthermore if the knots points are uniformly distributed, and the input is one dimensional.

4 Conclusion

A major research topic of soft computing area is the mathematical characterization of approaches in terms of their approximation property. Having proven that almost all soft computing techniques have the universal approximation property, researchers' interest turned to the determination of approximation rates and of constructive system designing methods satisfying a predefined approximation accuracy.

In this paper we derived optimal rate of convergence for α -cut based fuzzy interpolators. The achieved results exploited the connection of Shepard and KH interpolators. We determined for $\lambda > 2$ the saturation class and order for KH interpolator, and we got best approximation error estimates for $1 \leq \lambda \leq 2$ case.

References

- [1] R. E. Barnhill, R. P. Dube, and F. F. Little. Properties of Shepard's surfaces. *Rocky Mountain J. Math.*, 13:365–382, 1991.
- [2] E. K. Blum and L. K. Li. Approximation theory and feedforward networks. *Neural Networks*, 4(4):511–515, 1991.

- [3] J. L. Castro. Fuzzy logic controllers are universal approximators. *IEEE Trans. on SMC*, 25:629–635, 1995.
- [4] G. Crisculo and G. Mastroianni. Estimates of Shepard interpolatory procedure. *Acta Math. Hung.*, 61:79–91, 1993.
- [5] G. Cybenko. Approximation by superposition of sigmoidal functions. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.
- [6] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.
- [7] E. P. Klement, L. T. Kóczy, and B. Moser. Are fuzzy systems universal approximators? *Int. J. General Systems*, 28(2–3):259–282, 1999.
- [8] L. T. Kóczy and K. Hirota. Approximate reasoning by linear rule interpolation and general approximation. *Internat. J. Approx. Reason.*, 9:197–225, 1993.
- [9] L. T. Kóczy and K. Hirota. Ordering, distance and closeness of fuzzy sets. *Fuzzy Sets and Systems*, 60:281–293, 1993.
- [10] L. T. Kóczy and A. Zorat. Fuzzy systems and approximation. *Fuzzy Sets and Systems*, 85:203–222, 1997.
- [11] B. Kosko. Fuzzy systems as universal approximators. *IEEE Tr. on Computers*, 43(11):1329–1333, 1994.
- [12] V. Kůrková. Approximation of functions by perceptron networks with bounded number of hidden units. *Neural Networks*, 8(5):745–750, 1995.
- [13] D. Shepard. A two dimensional interpolation function for irregularly spaced data. In *Proc. of the 23rd ACM International Conference*, pages 517–524, 1968.
- [14] G. Somorjai. On a saturation problem. *Acta Math. Acad. Sci. Hungar.*, 32:377–381, 1978.
- [15] J. Szabados. On a problem of R. DeVore. *Acta Math. Acad. Sci. Hungar.*, 27:219–223, 1976.
- [16] J. Szabados. Direct and converse approximation theorems for Shepard operator. *J. Approx. Th. and its Appl.*, 7:63–76, 1991.
- [17] D. Tikk and P. Baranyi. Comprehensive analysis of a new fuzzy rule interpolation method. *IEEE Trans. on Fuzzy Systems*, 8(3):281–296, 2000.
- [18] D. Tikk, P. Baranyi, Y. Yam, and L. T. Kóczy. Stability of a new interpolation method. In *Proc. of the IEEE Int. Conf. on System, Man, and Cybernetics (IEEE SMC'99)*, volume III, pages 7–9, Tokyo, Japan, October, 1999.
- [19] D. Tikk, I. Joó, L. T. Kóczy, P. Várlaki, B. Moser, and T. D. Gedeon. Stability of interpolative fuzzy KH-controllers. *Fuzzy Sets and Systems*, 125(1):105–119, January 2002.
- [20] D. Tikk, L. T. Kóczy, and T. D. Gedeon. A survey on the universal approximation and its limits in soft computing techniques. Research Working Paper RWP-IT-01-2001, School of Information Technology, Murdoch University, Perth, W.A., 2001. p. 20.
- [21] L. X. Wang. Fuzzy systems are universal approximators. In *Proc. of the IEEE Int. Conf. on Fuzzy Systems*, pages 1163–1169, San Diego, 1992.
- [22] H. Ying. Sufficient conditions on uniform approximation of multivariate functions by general Takagi–Sugeno fuzzy systems with linear rule consequents. *IEEE Trans. on SMC, Part A*, 28(4):515–520, 1998.

Properties of fuzzy controller with conditionally firing rules

Mirko Navara, Jakub Št'astný

Center for Machine Perception, Department of Cybernetics,
Faculty of Electrical Engineering, Czech Technical University,
Technická 2, CZ-166 27 Praha, Czech Republic,
navara@cmp.felk.cvut.cz, stastnj1@fel.cvut.cz

Abstract. Studying thoroughly the mathematical principles of Mamdani–Assilian controller, we arrived at the conclusion that it represents the information given in the rule base efficiently, but in a somewhat distorted manner. Our objection is as follows: If we put an antecedent of one rule as the input, the output differs from the corresponding consequent. We may choose the rule base in such a way that the latter drawback disappears, but then we obtain also outputs that are not significant (i.e., the same as for the empty input). In [4] we observed that this behaviour is inevitable not only in Mamdani–Assilian controller, but also in any controller based on a compositional rule of inference, and we suggested a way to overcome this difficulty. We presented mathematical arguments why our controller should express the information contained in the rule base better than other approaches. In the present paper we describe a practical experiment that confirms the advantages of our controller. We also present its new description which should facilitate its application for other experts.

Keywords: Fuzzy controller, Compositional rule of inference, Firing rule, Fuzzy relational equation

1 Theoretical analysis of fuzzy controllers

Let X and Y denote the input and the output space, respectively. They are supposed to be convex subsets of finite-dimensional real vector spaces. For a set Z , we denote by $\mathcal{F}(Z)$ the set of all fuzzy subsets of Z . The expert's knowledge may be expressed by a base of rules of the form

$$\text{if } x \in X_i \text{ then } y \in Y_i ,$$

where $X_i \in \mathcal{F}(X)$ are *antecedents* and $Y_i \in \mathcal{F}(Y)$ are *consequents*, $i \in \{1, \dots, n\}$ [1]. Following [6], the knowledge from the rule base should be represented by a fuzzy relation $R \in \mathcal{F}(X \times Y)$. The Mamdani (or Mamdani–Assilian) controller [3] uses R in the form

$$R(x, y) = \max_{i \leq n} T(X_i(x), Y_i(y)) . \quad (1)$$

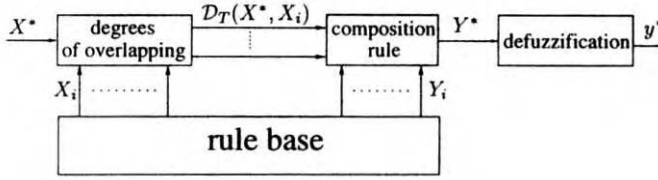


Figure 1: Block diagram of Mamdani–Assilian controller

Then we apply a *compositional rule of inference* which assigns to a fuzzy input $X^* \in \mathcal{F}(X)$ a fuzzy output $Y^* \in \mathcal{F}(Y)$ by $Y^* = X^* \circ_T R$, i.e.,

$$Y^*(y) = \sup_{x \in X} T(X^*(x), R(x, y)) , \quad (2)$$

where T is a fixed t-norm modelling a fuzzy conjunction [2]. Formulas (1), (2) can be combined to

$$\begin{aligned} Y^*(y) &= \sup_{x \in X} T\left(X^*(x), \max_{i \leq n} T(X_i(x), Y_i(y))\right) \\ &= \max_{i \leq n} T\left(\sup_{x \in X} T(X^*(x), X_i(x)), Y_i(y)\right) . \end{aligned}$$

The latter form allows for an effective calculation using the values

$$\mathcal{D}_T(X^*, X_i) = \sup_{x \in X} T(X^*(x), X_i(x)) , \quad (3)$$

called *degrees of overlapping* of X^*, X_i . The output is

$$Y^*(y) = \max_{i \leq n} T\left(\mathcal{D}_T(X^*, X_i), Y_i(y)\right) . \quad (4)$$

The block diagram of Mamdani–Assilian controller is shown in Fig. 1; the only formulas needed are (3) and (4).

Following [4], we suggest the following modifications of the Mamdani–Assilian controller:

1. Change the membership degrees in the input space by an increasing bijection $\rho: [0, 1] \rightarrow [0, 1]$, e.g., $\rho(t) = t^r$, $r > 1$. This changes the degree of overlapping $\mathcal{D}_T(X_i, X_j)$ to a (possibly smaller) value

$$\mathcal{D}_T(X_i \circ \rho, X_j \circ \rho) = \sup_{x \in X} T(\rho(X^*(x)), \rho(X_i(x))) . \quad (5)$$

2. Change the membership degrees in the output space by an increasing bijection $\sigma: [0, 1] \rightarrow [c, 1]$, where $0 \leq c < 1$, e.g., by the linear function $\sigma(t) = (1 - c)t + c$. To extend the inverse of σ to the whole interval $[0, 1]$, we use its *pseudoinverse*

$$\sigma^{[-1]}(t) = \begin{cases} \sigma^{-1}(t) & \text{if } t \geq c , \\ 0 & \text{otherwise .} \end{cases} \quad (6)$$

3. Replace the degree of overlapping in (4) with the *degree of conditional firing* of the i -th rule defined as

$$C_{T,i}(X^*) = \frac{\mathcal{D}_T(X^* \circ \rho, X_i \circ \rho)}{\max_{j \leq n} \mathcal{D}_T(X^* \circ \rho, X_j \circ \rho)}, \quad (7)$$

the output becomes

$$Y^*(y) = \sigma^{-1} \left(\max_{i \leq n} T(C_{T,i}(X^*), \sigma(Y_i(y))) \right). \quad (8)$$

The block diagram of the controller with conditionally firing rules is shown in Fig. 2; the formulas needed are (5), (6), (7) and (8). We do not deal here with defuzzification, leaving its influence to further study. Our basic observations refer to fuzzy subsets of the input and output space and they are independent of the method of defuzzification.

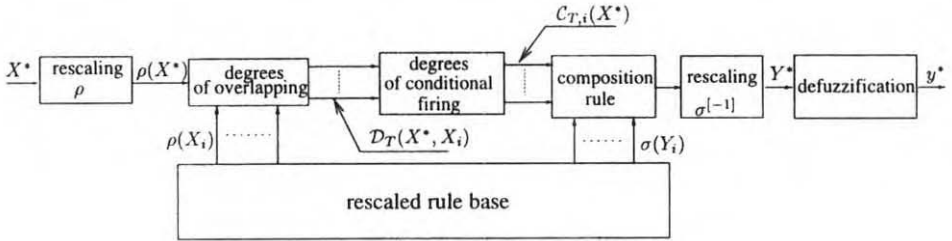


Figure 2: Block diagram of the controller with conditionally firing rules

Theorem 1. Let $(X_i, Y_i)_{i=1}^n$ be a rule base such that

- [C1] each X_i is normal, i.e., attains the value 1 at some point,
- [C2] $\inf_{x \in X} \max_{i \leq n} X_i(x) > 0$,
- [C3] $\forall i \in \{1, \dots, n\} : Y_i \not\leq \min_{j \neq i} Y_j$.

Let $\rho: [0, 1] \rightarrow [0, 1]$ be any increasing bijection such that

- [C4] $\exists c < 1 : i \neq j \implies \mathcal{D}_T(X_i \circ \rho, X_j \circ \rho) \leq c$.

Then for any increasing bijection $\sigma: [0, 1] \rightarrow [c, 1]$ the controller with conditionally firing rules (8) satisfies the following properties:

- [Int1] If the input coincides with the i -th antecedent, $X^* = X_i$, then the output coincides with the i -th consequent, $Y^* = Y_i$.
- [Int2] For each normal input X^* , the output Y^* is not contained in all consequents, i.e., $\exists i \in \{1, \dots, n\} : Y^* \not\leq Y_i$.
- [Int3] The output Y^* belongs to the convex hull of consequents Y_i of all rules such that $\exists x \in X : X_i(x) > 0, X^*(x) > 0$ (i.e., all firing rules).

Table 1: Comparison of Mamdani and CFR controller, experiment (a)

criterion	Mam. controller	CFR controller
asymptotic value [m] y_{∞}	-0.0021	0.0012
transient time [s]	3.56	3.05
cumulative quadratic error [m s]	0.0552	0.0569

Table 2: Comparison of Mamdani and CFR controller, experiment (b)

criterion	Mam. controller	CFR controller
maximum overshoot [m] σ	0.346	0.346
asymptotic value [m] y_{∞}	0.0051	0.0034
transient time [s]	14.8	11.1
cumulative quadratic error [m s]	0.583	0.441

We refer to [4] for the proof and analysis of the mathematical aspects of the latter theorem. It is also proved there that conditions [Int1]–[Int3] are almost contradictory for Mamdani–Assilian controller and typical shapes of membership functions. Contrary to this, assumptions of the latter theorem are quite weak and close to situations encountered in practice. Indeed, [C1] and [C2] just require that we have normal antecedents covering the whole input space and [C3] is quite a weak requirement on significance of consequents. [C4] represents disjointness of antecedents needed to distinguish between firing rules (not more than one rule may fire totally as this would cause a contradiction in the rule base). Once we find a bound $c < 1$ satisfying [C4], we have freedom in choosing σ with range $[c, 1]$.

2 Experiments and their results

Until now, we presented theoretical arguments for the controller with conditionally firing rules (CFR controller). To test its action in practice, we made simulated experiments on one of the classical tasks of control theory—balancing a ball on a plate (also called “ball on beam”). The aim is to achieve and stabilize a desired position of a ball by changing the angle of a plate on which it lies. The situation was simulated using Matlab/Simulink. The model included also static friction that has to be overcome to start movement. We made a series of tests under the following conditions:

- (a) nonzero initial position and zero initial velocity,
- (b) zero initial position and nonzero initial velocity,
- (c) zero initial position and zero initial velocity under presence of noise.

We first designed a Mamdani–Assilian controller solving the task. Then we adopted the algorithm as suggested above and compared the results. In experiment (a) CFR controller outperformed Mamdani–Assilian controller in almost all criteria, see Table 1. Its control was much more smooth, avoiding rapid changes of the control variable. This is particularly transparent from the details of position and velocity before reaching the steady state, see Fig. 3.

In experiment (b) we observed small oscillations of CFR controller around the desired value. We found out that this was due to a higher sensitivity together with a coarse discretization of time. For a finer time scale CFR controller again reached better parameters, see Table 2. In experiment (c) the situation was similar to (b): CFR controller performed worse

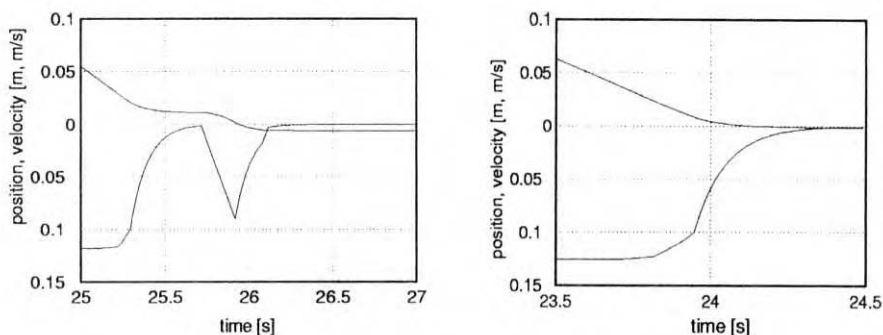


Figure 3: Outputs of Mamdani controller (left) and CFR controller (right) before reaching the steady state

for a coarse time discretization and better for a finer one. This requirement could be easily satisfied.

A detailed report on the method and results of tests can be found in [5] (in Czech only).

3 Conclusion

We derived and tested a new inference mechanism for fuzzy controllers. We proved theoretical arguments why our controller with conditionally firing rules expresses the knowledge from rule base better than Mamdani–Assilian controller in the sense of conditions [Int1]–[Int3]. Experimental results confirm this expectation. The controller with conditionally firing rules achieved smaller asymptotic value and transient time. Its action was much more smooth, avoiding unnecessary changes of sign of the control variable. Nevertheless, some problems with stability appeared if the discretization of time was too coarse. Further tests are expected to bring new experience and feedback for the use and development of the new controller.

Both the theoretical motivation and practical experiments suggest that the controller with conditionally firing rules allows us to enhance the performance in comparison to Mamdani–Assilian controller without a change of the rule base. Its application requires just adding three easy transformations. These require additional calculations, but do not extend the order of complexity. Our approach is now ready for further tests on other (more complex) practical control tasks.

References

- [1] Driankov, D., Hellendoorn, H., Reinfrank, M. (1993) *An Introduction to Fuzzy Control*. Springer, Heidelberg
- [2] Gottwald, S. (1993) *Fuzzy Sets and Fuzzy Logic*. Vieweg, Braunschweig
- [3] Mamdani, E. H., Assilian, S. (1975) An experiment in linguistic synthesis of fuzzy controllers. *Int. J. Man-Mach. Stud.*, 7, 1–13
- [4] Moser, B., Navara, M. (ND) Fuzzy controllers with conditionally firing rules. *IEEE Trans. Fuzzy Systems*, accepted
- [5] Št'astný, J. (2001) *Comparison of Mamdani and CFR Controller* (in Czech). Research Report CTU–CMP–2001–04, Center for Machine Perception, Czech Technical University, Prague, Czech Republic, ftp://cmp.felk.cvut.cz/pub/cmp/articles/navara/TR_Stastny.ps.gz

- [6] Zadeh, L. A. (1973) Outline of a new approach to the analysis of complex systems and decision processes. *IEEE Trans. Syst. Man Cybern.*, 3, 28–44

An Algorithm for the Network Steiner Tree Problem with Fuzzy Edge Lengths

Miloš Šeda

*Institute of Automation and Computer Science,
Faculty of Mechanical Engineering,
Brno University of Technology,
Technická 2, 616 69 Brno, Czech Republic
seda@uai.fme.vutbr.cz*

Abstract. In this paper, we deal with the Steiner tree problem (STP) on a graph in which a fuzzy number, instead of a real number, is assigned to each edge. We propose a modification of the shortest paths approximation based on the fuzzy shortest paths (FSP) evaluations. Since fuzzy min operation using the extension principle leads to nondominated solutions, we propose another approach to solving the FSP using Cheng's centroid point fuzzy ranking method.

Keywords: Steiner tree, Single shortest path problem, Fuzzy ranking, Binary heap, Priority queue

1 Introduction

The *Network Steiner tree problem* (NSTP) (or *Steiner tree problem in graphs*) [3], [7], [18] is concerned with connecting a subset of vertices at a minimal cost. More precisely, given an undirected connected graph $G = (V, E)$ with vertex set V , edge set E , nonnegative weights associated with the edges, and a subset B of V (called *terminals* or *customer vertices*), the problem is to find a subgraph T that connects the vertices in B so that the sum of the weights of the edges in T is minimised. It is obvious that the solution is always a tree and it is called a *Steiner minimum tree* for B in G .

Applications of the NSTP are frequently found in the layout of connection structures in networks and circuit design [1], [2], [4]. Their common feature is that of connecting together a set of terminals (communications sites or circuits components) by a network of the minimal total length.

If $|B| = 2$ then the problem is reduced to the *shortest path problem* and can be solved by Dijkstra's algorithm. In the case of $B = V$ the NSTP reduces to the *minimum spanning tree* (MST) *problem* and can be solved by Jarník's (Prim's), Borůvka's or Kruskal's algorithm. All these algorithms are polynomial. However, in the general case the NSTP is NP-complete [11], [14] and therefore it cannot be solved exactly for larger instances, i.e. heuristic or approximation methods must be used. Normally a Steiner minimum tree is not a minimum spanning tree only, it can also span some nonterminals, called *Steiner vertices*.

In the graphical representation, vertices and edges can correspond to locations and connections between locations, respectively. The weights (lengths or costs) of edges can express geographical distances of the corresponding vertices or transportation costs expended (or times spent) to move between their end vertices. While geographical distances can be stated deterministically, costs or times can fluctuate with traffic conditions, payload and so on. In the last two cases, deterministic values for representing the edge weights cannot be used. A typical way of expressing these uncertainties in the edge weights is to utilize fuzzy numbers based on fuzzy set theory. In this case, we must define an order relation between fuzzy numbers, because the fuzzy variant of the problem evaluates "fuzzy min" operations. As many approaches for the comparison of fuzzy numbers do not guarantee that fuzzy numbers are totally ordered, they lead to a number of nondominated paths (or Pareto Optimal paths). However, the number of nondominated trees derived from a large network can be too numerous, and it is difficult for a decision maker to choose a preferable tree.

In this paper, we propose a different approach based on Cheng's fuzzy ranking method [10] that makes it possible to solve the fuzzy Steiner minimum tree (FSMT) in a way similar to that of the crisp version of the problem.

2 Fuzzy ranking

Let us suppose that the weights of the edges are given by *linear triangular fuzzy numbers*. Mathematically, a linear triangular fuzzy number \tilde{A} can be represented by a triple (a_1, a_2, a_3) and its membership function $\mu_{\tilde{A}}$ is given by

$$\mu_{\tilde{A}}(x) = \begin{cases} 0 & , \text{ if } 0 \leq x \leq a_1 \\ \frac{x-a_1}{a_2-a_1} & , \text{ if } a_1 \leq x \leq a_2 \\ 1 & , \text{ if } x = a_2 \\ \frac{x-a_3}{a_2-a_3} & , \text{ if } a_2 \leq x \leq a_3 \\ 0 & , \text{ if } x \geq a_3 \end{cases} \quad (1)$$

The *addition* of these fuzzy numbers can be derived using Zadeh's extension principle and it is determined as follows:

$$\tilde{A} \oplus \tilde{B} = (a_1, a_2, a_3) \oplus (b_1, b_2, b_3) = (a_1 + b_1, a_2 + b_2, a_3 + b_3) \quad (2)$$

This operation always results in a triangular fuzzy number.

In the SPP we must also evaluate minimum operations. This means that it is necessary to have a method of ranking or comparison of fuzzy numbers. An *ordering relation* \preceq of fuzzy numbers can be defined, e.g., as follows:

$$\tilde{A} \preceq \tilde{B} \iff (a_1 \leq b_1) \wedge (a_2 \leq b_2) \wedge (a_3 \leq b_3) \quad (3)$$

However, this relation is not a complete ordering relation, as fuzzy numbers \tilde{A}, \tilde{B} satisfying $(\exists i, j \in \{1, 2, 3\}) : (a_i < b_i) \wedge (a_j > b_j)$ are not comparable by \preceq .

Let us consider fuzzy min operation defined as the fuzzy addition in the following way:

$$\tilde{\min}(\tilde{A}, \tilde{B}) = (\min(a_1, b_1), \min(a_2, b_2), \min(a_3, b_3)) \quad (4)$$

It is evident that for noncomparable fuzzy numbers \tilde{A}, \tilde{B} , this fuzzy min operation results in a fuzzy number different from both of them. For example, for $\tilde{A} = (5, 10, 13)$ and $\tilde{B} = (6, 9, 14)$, we get from equation (4) a fuzzy min $(5, 9, 13)$ which differs from \tilde{A} and \tilde{B} .

The previous remarks demonstrate the difficulties with comparisons of fuzzy numbers. For this reason, the ranking or ordering methods of fuzzy quantities have been proposed by many authors. Most of them were summarized in [9, 16, 17]. Further methods can be found, e.g., in [5, 6, 8, 10, 12, 15, 19]. Unfortunately, none of these methods is commonly accepted.

In this paper, we use for simplicity the fuzzy ranking method described in [10], modified for the case of triangular fuzzy numbers. This method uses inverse functions $g_A^L : [0, 1] \rightarrow [a_1, a_2]$ and $g_A^R : [0, 1] \rightarrow [a_2, a_3]$ derived from functions $f_A^L : [a_1, a_2] \rightarrow [0, 1]$ and $f_A^R : [a_2, a_3] \rightarrow [0, 1]$, respectively.

From $y = \frac{x-a_1}{a_2-a_1}$ and $y = \frac{x-a_3}{a_2-a_3}$ we can easily derive that

$$g_A^L = a_1 + (a_2 - a_1)y, \quad g_A^R = a_3 + (a_2 - a_3)y \quad (5)$$

The ranking function is defined as a distance between the *centroid point* $(\tilde{x}_0, \tilde{y}_0)$ and the origin, i.e.

$$R(\tilde{A}) = \sqrt{(\tilde{x}_0)^2 + (\tilde{y}_0)^2} \quad (6)$$

where

$$\tilde{x}_0 = \frac{\int_{\text{Supp } \tilde{A}} x \mu_{\tilde{A}}(x) dx}{\int_{\text{Supp } \tilde{A}} \mu_{\tilde{A}}(x) dx}, \quad \tilde{y}_0 = \frac{\int_0^1 (y g_A^L) dy + \int_0^1 (y g_A^R) dy}{\int_0^1 (g_A^L) dy + \int_0^1 (g_A^R) dy} \quad (7)$$

and $\text{Supp } \tilde{A}$ is the support of \tilde{A} .

Fuzzy numbers \tilde{A}, \tilde{B} are then ranked by their ranking function values $R(\tilde{A})$ and $R(\tilde{B})$.

3 Shortest paths approximation

The shortest paths approximation [3] is an analogy of Jarník's algorithm for finding a minimum spanning tree. We will describe it in a pseudopascal code.

```

input:  connected undirected graph  $G = (V, E)$ ,
        weight function,
        set of terminals  $B \subset V$ 
output: Steiner tree  $T$  for  $B$ 
select arbitrary terminal in  $B$  and denote it  $x_1$ 
 $T := \{x_1\}$ ;
 $k := 1$ ;
while  $k < |B|$  do
    begin
        determine a terminal  $x_{k+1} \notin T$  closest to  $T$ 
         $T := T \cup \{x_{k+1}\}$ ;
         $T := T \cup \{\text{the shortest path joining } T \text{ and } x_{k+1}\}$ ;
         $k := k + 1$ 
    end;
 $\{T \text{ is an approximation of the SMT}\}$ 

```

The solution can be improved by two additional steps:

1. Determine a fuzzy minimum spanning tree for the subnetwork of G induced by the vertices in T .
2. Delete from this minimum spanning tree nonterminals of degree 1 (one at a time) and edges incident with them. The resulting tree is the (suboptimal) solution.

3.1 Fuzzy SPP algorithm

Finding the shortest path from a specified *source* (or an *origin*) to a *sink* (or a *destination*) is a fundamental problem in transportation, routing, and communications applications. Alternatively, the problem can be formulated as finding the shortest paths from a source to all other locations (communication sites). The computational problem is called the *single source shortest path problem* (SPP for short).

Before we discuss the algorithm for the fuzzy SPP, we will briefly summarize the data structures used in it.

3.1.1 Implementation

The most important data structure substantially determining the efficiency is the *priority queue* [13]. The priority queue (Q) supports these operations:

- *Insert*(Q, u, key) - insert u with the key value key in Q ,
- *ExtractMin*(Q) - extract the item with the minimum key value in Q , and
- *DecreaseKey*(Q, u, new_key) - decrease the value of u 's key value to new_key .

The priority queue can be easily implemented by a *binary heap*. This is a binary tree with vertices numbered by integers and satisfying these conditions: (1) Each vertex of a binary heap, which does not lie in the last two levels has two successors; (2) In the last level, all vertices are placed from the left. This means that if we pass vertices in the last but one level from left to right, then some of them (or none) have two successors, then at the most one vertex may exist with one successor and all other vertices of this level are leaves, and (3) The number of each vertex is not higher than the numbers of its successors.

The binary tree can be represented by a one-dimensional array. It can be proved that for the defined numbering of the binary heap vertices the j th element in the i th level of a binary heap corresponds to position $2^{i-1} + j - 1$ of the array; left and right successors of vertex i have positions $2i$ and $2i + 1$, respectively, and its predecessor has position $\lfloor i/2 \rfloor$ (that is position $i \div 2$ in the Pascal notation).

We will skip detailed descriptions of Insert, ExtractMin and DecreaseKey operations because they can be found in all books dealing with algorithms and data structures (e.g. [11]), only noting that the time complexity of all of them is $O(\log n)$ where n is the number of vertices.

3.1.2 Algorithm for the fuzzy SPP

Dijkstra's algorithm for the "deterministic" SPP finds the shortest paths from a source s to all other vertices. Now we will generalise it for the case of fuzzy edge lengths. Let $\tilde{d}[v]$ be the fuzzy number corresponding to the length of the shortest path from s to v . Initially, $\tilde{d}[s] = (0, 0, 0)$ and all other $\tilde{d}[v]$ values are set to (∞, ∞, ∞) . The algorithm is based on gradual improvements of the shortest paths distance from s to the other vertices. Let us consider an edge (u, v) whose weight is $\tilde{w}(u, v)$ and suppose that we have already computed current estimates of $\tilde{d}[u]$, $\tilde{d}[v]$. If $R(\tilde{d}[u] \oplus \tilde{w}(u, v)) < R(\tilde{d}[v])$, then $\tilde{d}[u] \oplus \tilde{w}(u, v)$ becomes a new estimate of $\tilde{d}[v]$. The process by which an estimate is updated is called *relaxation*. The shortest way back to the source is through u by updating the predecessor pointer. If we repeat the

relaxation for all edges then $\tilde{d}[v]$ values converge to the shortest paths of vertices v to the source. Let S be a set of vertices for which we know the final value $\tilde{d}[v]$. Initially, S is empty. The question is, how do we select which vertex of $V - S$ to add next to S ? The algorithm uses a greedy strategy, i.e. it takes the vertex for which $\tilde{d}[u]$ is minimum, in other words, it takes the unprocessed vertex that is closest (by our estimate) to s .

In order to perform this selection efficiently, we store the vertices of $V - S$ in a priority queue (binary heap), where the key value of each vertex u is $\tilde{d}[u]$. Information on which vertices are in S (they have final value $\tilde{d}[v]$) is stored in an array of Boolean variables *determined*. Here is the algorithm.

```

Input           : connected weighted graph  $G = (V, E)$  with fuzzy edge lengths
                   $\tilde{w}(e), e \in E$ 
                   $s$  - source (root);
output          :  $\tilde{d}[u]$ ;
auxiliary variables :  $Adj[u]$  - set of neighbours of vertex  $u$ .
for  $\forall u \in V$  do
  begin  $\tilde{d}[u] := (0, \infty, \infty)$ ;
         $determined[u] := \text{False}$ ;
         $pred[u] := \text{nil}$ 
  end;
 $\tilde{d}[s] := (0, 0, 0)$ ;
 $Q := \text{priority\_queue}(V)$ ; { push all vertices into  $Q$  ordered by  $\tilde{d}[u]$  }
while  $NonEmpty(Q)$  do
  begin  $u := ExtractMin(Q)$ ;
        for  $\forall v \in Adj[u]$  do
          if  $R(\tilde{d}[u] \oplus \tilde{w}(u, v)) < R(\tilde{d}[v])$ 
            then begin  $\tilde{d}[v] := \tilde{d}[u] \oplus \tilde{w}(u, v)$ ;
                       $DecreaseKey(Q, v, \tilde{d}[v])$ ;
                       $pred[v] := u$ 
            end;
           $determined[u] := \text{True}$ 
        end; { Fuzzy Dijkstra }
  { pointers in array  $pred$  define the inverted tree of the shortest paths pointing back to  $s$  }

```

4 Analysis of the algorithm

In this section, we analyse the time complexity of the proposed algorithm for the fuzzy network Steiner tree problem.

Theorem 1. *The proposed algorithm runs in $O(|B||E| \log |V|)$ time.*

Proof. Since the time complexity of the algorithm is given by $|B| \times$ (time complexity of the fuzzy SPP algorithm), we can focus on the FSSP. Let $O(T_R)$ be the time of the centroid point evaluation. To analyze the algorithm, we account for the time spent on each vertex as it is extracted from the priority queue. It takes $O(\log |V|)$ time to extract this vertex from the queue. For each incident edge, we spend potentially $O(\log |V|)$ time decreasing the key of the neighbouring vertex. Thus the time is $O(\log |V| + T_R \deg(u) \log |V|)$. The other steps of

the update are constant time. So the overall running time is

$$\begin{aligned} T(V, E) &= \sum_{u \in V} (\log |V| + T_R \deg(u) \log |V|) = \sum_{u \in V} (1 + T_R \deg(u)) \log |V| = \\ &= \log |V| \sum_{u \in V} (1 + T_R \deg(u)) = \log |V| (|V| + T_R \cdot 2|E|) = O((|V| + |E|) \log |V|) \end{aligned}$$

Since G is connected, $|V|$ is asymptotically no greater than $|E|$, so this is $O(|E| \log |V|)$. •

5 Conclusion and future work

In this paper, the problem of finding the network Steiner minimum tree was studied. In contrast with traditional approaches, we supposed that the weights of edges were given by linear triangular fuzzy numbers and proposed a fuzzy modification of the shortest paths approximation with an emphasis on an efficient implementation of the shortest path subroutine using the priority queue. For comparisons of fuzzy numbers, Cheng's centroid point fuzzy ranking method was used.

In future, we intend to investigate a different approach based on a reformulation of the problem where the NSTP is to find $S \subseteq V$ so that the spanning tree of the subgraph of G induced by $B \cup S$ has a minimum total weight. The solutions in the search space are then restricted to binary strings where a 1 or 0 corresponds to whether or not a vertex from $V - B$ is included in the Steiner tree. Because of the exponentially growing size of the search space we will use metaheuristics (such genetic algorithms and simulated annealing) for finding an approximation of the solution.

This paper was supported by the research project CEZ: J22/98: 261100009 "Non-traditional methods for investigating complex and vague systems".

References

- [1] D. Cavendish, A. Fei, M. Gerla, and R. Rom. On the Construction of Low Cost Multicast Trees with Bandwidth Reservation. In *Proceedings of the International Conference on High-Performance Computing and Networking HPCN*, Amsterdam, 1998. 29 pp.
- [2] D.W. Corne, M.J. Oates, and G.D. Smith (eds.). *Telecommunications Optimization: Heuristic and Adaptive Techniques*. John Wiley & Sons, New York, 2000. ISBN 0-471-98855-3.
- [3] D.-Z. Du, J.M. Smith, and J.H. Rubinstein. *Advances in Steiner Trees*. Kluwer Academic Publishers, Dordrecht, 2000. ISBN 0-7923-6110-5.
- [4] A. Fink, G. Schneidereit, and S. Voss. Solving General Ring Network Design Problems by Meta-Heuristics, chapter in: Laguna, M. and González Velarde, J.L. (eds.): *Computing Tools for Modeling, Optimization and Simulation (Interfaces in Computer Science and Operations Research)*, pages 91–113. Kluwer Academic Publishers, Boston, 1999.
- [5] P. Fortemps and M. Roubens. Ranking and Defuzzification Methods Based on Area Compensation. *Fuzzy Sets and Systems*, 82:319–330, 1996.
- [6] P. Grzegorzewski. Metrics and Orders in Space of Fuzzy Numbers. *Fuzzy Sets and Systems*, 97:83–94, 1998.
- [7] F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner Tree Problem*. North-Holland, Amsterdam, 1992. ISBN 0-444-89098-X.

- [8] S. Chanas and P. Zielinski. Ranking Fuzzy Interval Numbers in the Setting of Random Sets - Further Results. *Information Sciences*, 117:191–200, 1999.
- [9] P.-T. Chang and E.S. Lee. *Fuzzy Arithmetics and Comparison of Fuzzy Numbers*, chapter in: Delgado, M., Kacprzyk, J., Verdegay, J.-L. and Vila, M.A. (eds.): *Fuzzy Optimization*, pages 69–82. Physica-Verlag, Heidelberg, 1994.
- [10] C.-H. Cheng. A New Approach for Ranking Fuzzy Numbers by Distance Method. *Fuzzy Sets and Systems*, 95:307–317, 1998.
- [11] S. Khuller. Design and Analysis of Algorithms. Lecture Notes, University of Maryland, Department of Computer Science, 1994. 112 pp.
- [12] M. Modarres and S. Sadi-Nezhad. Ranking Fuzzy Numbers by Preference Ratio. *Fuzzy Sets and Systems*, 118:429–436, 2001.
- [13] D.M. Mount. Design and Analysis of Computer Algorithms. Lecture Notes, University of Maryland, College Park, 1999. 131 pp.
- [14] J. Plesník. *Grafové algoritmy*. VEDA, vydavateľstvo Slovenskej akadémie vied, Bratislava, 1983.
- [15] A. Sengupta and T.K. Pal. On Comparing Interval Numbers. *European Journal of Operational Research*, 127:28–43, 2000.
- [16] X. Wang and E.E. Kerre. Reasonable Properties for Ordering of Fuzzy Quantities (I). *Fuzzy Sets and Systems*, 118:375–385, 2001.
- [17] X. Wang and E.E. Kerre. Reasonable Properties for Ordering of Fuzzy Quantities (II). *Fuzzy Sets and Systems*, 118:387–405, 2001.
- [18] P. Winter. The Steiner Problem in Networks: A Survey. *Networks*, 17:129–167, 1987.
- [19] J.-S. Yao and K. Wu. Ranking Fuzzy Numbers Based on Decomposition Principle and Signed Distance. *Fuzzy Sets and Systems*, 116:275–288, 2000.

A Method For Learning of Hierarchical Fuzzy Systems

R. NOWICKI, R. SCHERER, L. RUTKOWSKI

Department of Computer Engineering,

Technical University of Czestochowa,

Al. Armii Krajowej 36, 42-200 Czestochowa, Poland.

{rmowicki, rscherer, lrutko}@kik.pcz.czest.pl

Abstract. This paper presents a new way of learning in hierarchical fuzzy systems. The intermediate variables have always the same physical quantity as the output variable in the last stage. The most important variables are placed in the first stage. Variable importance is determined from the training data. Moreover a short introduction to hierarchical systems is given.

Keywords: *neuro-fuzzy systems, curse of dimensionality, hierarchical fuzzy systems*

1. Introduction

Fuzzy systems have been successfully applied in many domains of science and engineering. Most of them utilize the single stage reasoning, i.e. there are only the input variables, applied to the input of the system, and the output variables as an outcome of a reasoning process. They are eagerly used by engineers because the expert knowledge in the form of fuzzy rules with linguistic terms is comprehensible and easily interpretable. The most popular of them are Mamdani and TSK systems, which served as the basis of numerous network structures developed so far. Such fuzzy-neural networks have the ability to learn, thanks to their connectionist architecture. The rules, the shapes of membership functions and their positions can be adjusted through supervised or unsupervised training. However, when the number of input variables is high, the resultant fuzzy system is extremely complex. When the whole input space has to be covered by fuzzy rules, each combination of input linguistic terms constitutes a fuzzy rule. Having given n inputs and m linguistic terms for every input, we have m^n possible rules. It shows that the number of fuzzy rules increases exponentially with the number of inputs. This phenomenon is called the curse of dimensionality. It causes multidimensional systems to have an enormous number of adjustable parameters. Learning of such systems is inconvenient because of a large number of local minima, computational complexity and memory requirements.

Hierarchical fuzzy systems use multi-stage fuzzy logic inference, and hence three kinds of linguistic variables are involved, namely: input, intermediate and output variables. This kind of reasoning is similar to human thinking. We would choose the most important input variables to be placed in the first stage and the next important ones in the consecutive stages.

Hierarchical fuzzy systems have been first introduced by Raju et al. in 1994, as a method to overcome the curse of dimensionality. The system was composed of several low-dimensional fuzzy systems, connected in a hierarchical way. There was only one system in each stage. It was proved that the overall number of fuzzy rules is minimal when each stage has only two inputs.

From the structural point of view, hierarchical systems can be divided into three groups [1][2][3] (Fig. 1). In aggregated hierarchical fuzzy systems, the lowest level serves as an input of the entire structure. Outputs of this first level are connected to the second stage inputs and so on, until the highest level whose outputs become the outputs of the whole structure. Because this structure is a combination of several single stage fuzzy systems, the number of rules is smaller than in single stage systems.

In incremental hierarchical fuzzy system, outputs of subsystems from the previous stage are also fed to the one of the next stage subsystem input, but in every stage, there is only one subsystem. The subsystems have one of their inputs connected to the previous stage output and the next input (or inputs) serves as one of the inputs of the whole structure. Thus, every stage has one or more input variables connected, until all input variables are used. Wang [13] proved that the incremental hierarchical systems are universal approximators and analysed sensitivity of particular inputs.

Duan et al. [4] used directly the notion of syllogistic reasoning, cascading fuzzy neural networks in two stages (Fig. 1c). The second stage network has its inputs connected to the outputs of the first stage. In this way, one decision is made entirely on the basis of another decision. The whole system is trained by hybrid learning algorithm, i.e. the optimal rule base is searched by genetic algorithm and then parameters are fine-tuned by backpropagation.

Because it is hard to obtain the hierarchical rule base, there were also several genetic approaches to hierarchical rule base design [6][7][5][11].

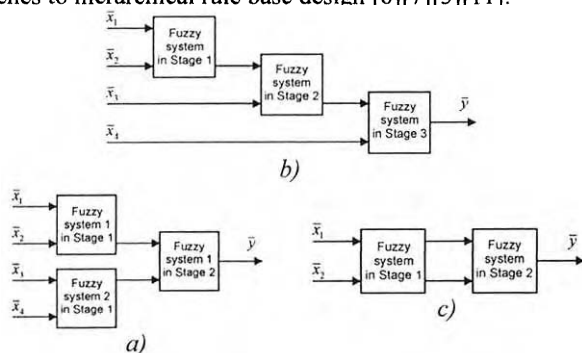


Figure 1. Examples of hierarchical fuzzy systems: a) aggregated, c) incremental, c) cascaded.

The intermediate variables can be fuzzy or crisp. Using the fuzzy ones is intuitive and straightforward, although is computationally demanding. We must determine an intersection of the input fuzzy set and the fuzzy set in IF-part of a rule. Then, we have to find the maximum of this resultant fuzzy set. Crisp intermediate variables are easier to handle, but they require defuzzification and fuzzification, and we miss the information carried by fuzzy signals.

2. New Method of Learning

In the above-mentioned structures, all intermediate variables are different quantities. When a structure does not reflect the real system hierarchy, intermediate variables do not have a physical meaning. Then determining a fuzzy rule base by an expert is very difficult. To overcome this disadvantage, we propose a hierarchical structure in which individual subsystem outputs have the same physical quantity (Fig. 2). The subsystem in the first stage gives a coarse result and the following stages give gradually a finer one, having given information from the successive inputs. Subsystems are trained by the same desired

response. In the backpropagation training, every subsystem is trained separately, beginning from the lowest level, minimizing the error as the difference between the desired output of the whole system and the real output of the subsystem. Then the next subsystem is trained, and so on up to the highest level. Error signal does not need to be backpropagated through the entire structure, so there are no complicated derivatives and there are less local minima.

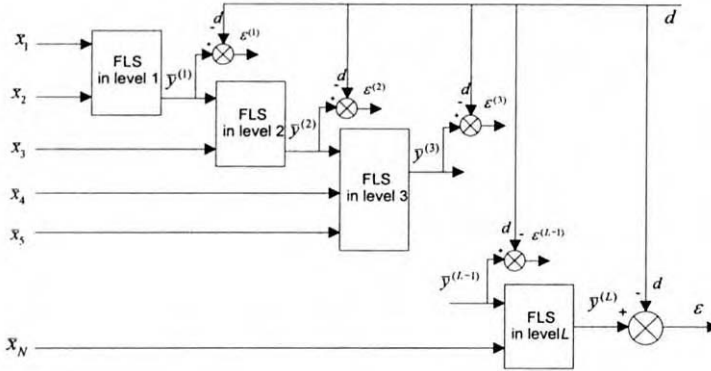


Figure 2. Incremental hierarchical fuzzy systems with intermediate variables being the same physical quantity

Each kind of fuzzy systems can be used as a subsystem. For demonstration we used Mamdani subsystems with Gaussian antecedent functions.

3. Input Selection

Usually a model is designed using a training data set. The data set allows approximating the input-output mapping. Real world applications and problems involve sometimes many inputs, sometimes even hundreds. State of the system output depends on all variables but not to the same degree. That is why we sometimes need to choose only the most important variables. We can obtain a simpler model and reduce the computational burden. Moreover, determining the importance of each input variable makes it possible to assign inputs according to their importance to consecutive levels of hierarchical systems or to choose only the most important variables. There are many methods for influential variables identification.

A straightforward and intuitive method is presented in [2]. Having given a data set $Z = \{z_1, K, z_D\}$ where $z^d = \{x^d, y^d\}$, $d = 1, K, D$, $x^d = [x_1^d, x_2^d, K, x_N^d]^T$, $y^d = [y_1^d, y_2^d, K, y_M^d]^T$ we can compute the degree to which a deviation of the input x_n causes a deviation of y_m . In this method, we directly obtain the importance of each input variable.

Any two output values y_d and y_j can be approximated using Taylor series expansion in a fixed point $[\bar{x}_1, \bar{x}_2, K, \bar{x}_N]$

$$y_d = f(\bar{x}_1, \bar{x}_2, K, \bar{x}_N) + \sum_{n=1}^N \left. \frac{f}{x_n} \right|_{x_n=\bar{x}_n} (x_n^d - \bar{x}_n) + r_d \quad (1)$$

$$y_j = f(\bar{x}_1, \bar{x}_2, K, \bar{x}_N) + \sum_{n=1}^N \left. \frac{f}{x_n} \right|_{x_n=\bar{x}_n} (x_n^d - \bar{x}_n) + r_j \quad (2)$$

where r_d and r_j are higher order residuals and can be neglected. By subtracting (1) from (2) we get

$$y_d - y_j = \sum_{n=1}^N b_n (x_n^d - x_n^j) \quad (3)$$

where

$$b_n = \left. \frac{f}{x_n} \right|_{x_n=\bar{x}_n} \quad (4)$$

Taking every pair of vectors \mathbf{z}^d and \mathbf{z}^j , in case of D data vectors, we will have C_2^D combinations. In real data sets, it gives a large number of possible combinations; so we can randomly choose a smaller number of pairs, say q . Then we construct matrices with input and output data pairs differences, bounded together with vector of importance degrees

$$\mathbf{y}_v = \mathbf{X}_v \mathbf{b} \quad (5)$$

where \mathbf{y}_v is $q \times 1$ vector of output data pairs, \mathbf{X}_v is $q \times N$ matrix of input data pairs and \mathbf{b} is $N \times 1$ vector of unknown values determining importance of each input x_n . Because the most frequent situation is when a data set is much longer than the number of inputs, Eq. (2) is over-determined and there is no exact solution. Estimated value of \mathbf{b} can be found by least squares estimation (LSE), yielding the following equation:

$$\mathbf{b} = (\mathbf{X}_D^T \mathbf{X}_D)^{-1} \mathbf{X}_D^T \mathbf{y}_D \quad (6)$$

The elements of vector \mathbf{b} are importance degrees of the inputs and can be of any value. Normalized degree values $s_j \in [0, 1]$ can be obtained by

$$s_j = \frac{|b_j|}{\sum_{n=1}^N |b_n|} \quad (7)$$

Data set must be normalized before using the above method.

4. Numerical Simulations

We tested our approach on the function [5]

$$y = \frac{4x_1x_2 + x_3x_4 + x_5 + 2\sin(\rho x_6)}{8} \quad (8)$$

We obtained 500 samples and 100 of them were used for learning and the whole data set for testing. Using the method from section 3, we computed input significance (Table 1).

Input number	Significance
1	0.2522
2	0.2637
3	0.1325
4	0.1653
5	0.1565
6	0.0298

Table 1. Input significance

Every subsystem is a standard Mamdani fuzzy system with max-product inference, i.e. product T-norm used instead of fuzzy implication, algebraic product as Cartesian product, singleton consequents and center average defuzzifier. Gaussian functions were used in the rule antecedents. Every subsystem had 6 linguistic variables for every input. The mean squared error is shown in Table 2. The first row shows error for significant variables placed in the first stages and less important variables placed in the last stages. The second row shows the opposite situation, where variables are deployed in reverse order.

MSE	Stage 1	Stage 2	Stage 3	Stage 4	Stage 5
The most important variables placed in the first stages	0,00964	0,00780	0,00731	0,00700	0,00685
The most important variables placed in the last stages	0,02162	0,02234	0,02186	0,01624	0,00978

Table 2. Testing mean squared error after 100 000 iterations at the outputs of each stage for two configurations of significant variables deployment.

5. Conclusion

In this paper we develop a new way of a hierarchical fuzzy system learning. An error signal does not have to be propagated through the whole hierarchical structure. Intermediate variables are crisp, so fuzziness does not spread during multiple consecutive fuzzy reasonings. Inputs can be prioritized and every stage can make more accurate decision thanks to new data from its inputs and an output from the previous stage. The obtained results are consistent with our intuitions and beliefs that placing the most important variables at the beginning increases performance. This is similar to human thinking where the most important decisions are made first.

References

- [1] Chung F.-L., Duan J.-C., Deriving Multistage FNN Models From Takagi and Sugeno's Fuzzy Systems, Proceedings of 1998 IEEE World Congress on Computational Intelligence, FUZZ-IEEE (1998), pp. 1259-1264.
- [2] Chung F.-L., Duan J.-C., On Multistage Fuzzy Neural Network Modeling, IEEE Transactions On Fuzzy Systems, Vol. 8, No. 2, April 2000.
- [3] Duan J.-C., Chung F.-L., A Mamdani Type Multistage Fuzzy Neural Network Model, Proceedings of 1998 IEEE World Congress on Computational Intelligence, FUZZ-IEEE (1998), pp.1253-1258.

- [4] Duan J.-C., Chung F.-L., Cascading Fuzzy Neural Networks, Proceedings of 1999 IEEE International Fuzzy Systems Conference Proceedings, Seoul, Korea (1999), pp. I-55-I-60.
- [5] Fukuda T., Hasegawa Y., Shimojima K., Structure Organization of Hierarchical Fuzzy Model using by Genetic Algorithm, Proc. of FUZZ-IEEE/IFES'95, Vol.1, (1995), pp.295-299.
- [6] Hoffman F., Pfister G., Automatic Design of Hierarchical Fuzzy Controllers Using Genetic Algorithms, 2nd European Congress on Intelligent Techniques and Soft Computing (EUFIT '94) (1994), Aachen.
- [7] Hoffman F., Pfister G., A New Learning Method for the Design of Hierarchical Fuzzy Controllers Using Messy Genetic Algorithms, Sixth International Fuzzy Systems Association World Congress (IFSA'95), vol.1 (1995), pp.249-252, Sao Paulo.
- [8] Maeda H., Yonekura H., Nobusada Y., Murakami S., Study on the Spread of Fuzziness in Multi-Stage Approximate Reasoning, Proceedings of IEEE Int. Conf. on Fuzzy Systems – FUZZ-IEEE'95, Yokohama, Japan (1995), pp. 1455-1460.
- [9] Nowicki R., Scherer R., Hierarchical Fuzzy System With A New Approach To Transferring The Intermediate Variables, Proceedings of The 10th International Conference on Systems Modelling Control SMC 2001, pp. 103-108.
- [10] Raju G.V.S., Zhou J., Kisner R.A., Hierarchical fuzzy control, in: Advances in Intelligent Control, Taylor & Francis Ltd, 1994, pp. 243-258.
- [11] Shimojima K., Fukuda T., Hasegawa Y., Self-tuning fuzzy modelling with adaptive membership function, rules, and hierarchical structure based on genetic algorithm, Fuzzy Set and Systems, vol. 71, (1995), pp. 295-309.
- [12] Wang L.-X., Analysis and Design of Hierarchical Fuzzy Systems, IEEE Transactions on Fuzzy Systems, vol. 7, no. 5, (1999), October.
- [13] Wang L.-X., Universal approximation by hierarchical fuzzy systems, Fuzzy Sets and Systems 93 (1998), pp. 223-230.

Risk Analysis in Fuzzy Flow Networks

Roman V. TYSHCHUK

Information Systems Department, AMI Corporation, Donetsk, Ukraine
rt_science@hotmail.com

Abstract. In this paper the problem of the inconsistency situation appearance in the network with fuzzy flows is shown. The method of risk analysis is presented and the maximum network flow algorithm with fuzzy capacities and predetermined risk level is proposed. These algorithmic solutions are demonstrated by examples.

Keywords: Fuzzy networks; Fuzzy flows; Risk analysis

1. Introduction and Problem Definition

In the maximum flow problem we are given a flow network $G = (V, E)$, which is oriented graph with two distinguished vertices S and T , where S is called the source and T the sink. With every arc $e \subseteq E$ of a flow network is associated a capacity $C(e) \geq 0$. A flow is a function $f: E \rightarrow \mathbb{R}^+ \cup \{0\}$ that satisfies the following two constraints:

$$\text{For all } e \subseteq E: f(e) \leq c(e) \quad (1)$$

$$\text{For each } v \subseteq V, v \notin \{S, T\}: \sum_{(i,v) \in E} f(i,v) = \sum_{(v,j) \in E} f(v,j) \quad (2)$$

There are many reasons why the uncertainty may appear in the flow networks, such as measurement errors, possibility to define the network characteristic value only within the bounds of some range and others. There are several ways in which a network can be fuzzy [1,2]. One of much interest and an important type of network fuzziness occurs when the network has known vertices and arcs, but fuzzy weights (or capacities) on the arcs.

The interpretation of the capacity constraint (1) is one of the main difficulties in the max flow problem in fuzzy networks. And in a fuzzy case it is essential to find a workable analogue to the crisp inequality $f(e) \leq c(e)$. To achieve the given inequality it is necessary to define $\tilde{f}(e) \leq \tilde{c}(e)$ as follows [3]:

$$\tilde{f}(e) \leq \tilde{c}(e) \text{ if } \forall i: \underline{f}_i(e) \leq \underline{c}_i(e) \text{ \& } \bar{f}_i(e) \leq \bar{c}_i(e), \quad (3)$$

where i - is the α -cut of the fuzzy number.

Thus during the solving of the max flow problem in fuzzy networks in some cases failures may take place because the max flow value in the arc in a real situation (as a crisp value) may be greater than the capacity of this arc in concrete time (also as a crisp value). Therefore it is necessary to define a risk of the inconsistency situation appearance in fuzzy networks and then to define a fuzzy maximum flow in the network with predetermined risk value. This task will be considered with (L-R) -type fuzzy numbers, which are the most commonly used in applications.

2. Mathematical Model of the Risk Analysis Methods

2.1 Risk constraint defining principle

One of the approaches to ranking fuzzy numbers [4] is the method assuming that the decision maker a priori chooses a degree of conformity for which the inequality may be considered as true by himself. To define a risk value of the inconsistency situation appearance it is possible to use one of the four different ways describing the relative position of two fuzzy numbers from the point of view of possibility theory [5] - an integrated superiority coefficient PSE. By defining PSE for every arc in the network we define the possibility of the fact that the greatest values of the flow $\tilde{f}(e)$ will be greater than the lowest values of the capacity $\tilde{c}(e)$.

The PSE coefficient is calculated using a classical formula for (L-R)-type fuzzy numbers:

$$\text{PSE} = \max(0, \min(1, 1 + (\tilde{f} - \underline{c})/(\beta + \gamma))), \quad (4)$$

where $\tilde{f} = (\tilde{f}, \underline{f}, \alpha, \beta)_{LR}$, $\tilde{c} = (\tilde{c}, \underline{c}, \gamma, \delta)_{LR}$. If $\beta + \gamma = 0$ then it is necessary to define β and γ as value $> 1/2 \min(|\underline{f} - \underline{c}|, |\underline{f} - \tilde{c}|, |\tilde{f} - \underline{c}|, |\tilde{f} - \tilde{c}|)$. If $\tilde{f} = \underline{c}$ and $\beta + \gamma = 0$ then it is possible to replace β and γ by any positive numbers.

Then the total risk coefficient of the inconsistency situation appearance in the whole network should to be defined as following:

$$\text{RISK}_{\text{total}} = \max(\text{PSE}(e)) \quad (5)$$

Thereupon the maximum flow problem with predetermined risk level should to be considered. This problem will be described in the case with integer values of the components of fuzzy capacities and fuzzy flows in the network.

The main idea of the maximum network flow algorithm in fuzzy networks with predetermined risk level is following. For all arcs in the network the new capacities should be defined to satisfy the risk constraint. On the basis of the PSE formula it is necessary to set $\tilde{f} < \underline{c}$ if the predetermined risk level $0 < \text{Risk} < 1$. In this case (L-R)-type fuzzy number which represents a flow value will become a triangular fuzzy number where $\tilde{c}_{\text{new}} = \underline{c}_{\text{new}}$. To find a new capacity it is necessary to decrease in the first loop a $\tilde{c}_{\text{new}} = \underline{c}_{\text{new}}$ value and in the internal loop it is necessary to decrease a δ value (Fig.1).

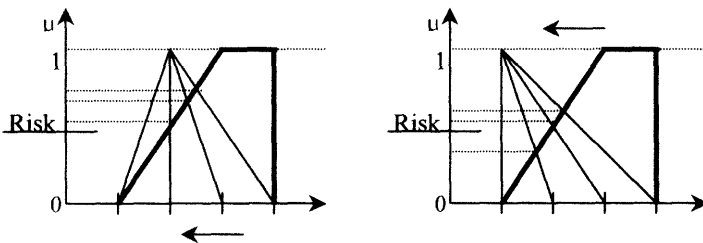


Figure 1. Risk constraint defining principle.

After each decreasing step it is necessary to compute a PSE value and also check if the new fuzzy number is correct. After the new capacities definition the maximum flow algorithm with fuzzy input data should to be used [6].

2.2 Algorithm

Algorithm

For each arc in the network do

If $\underline{c} - 1 \geq 0$ then

If $\gamma = 0$ then

$$\tilde{c}_{new} = (\underline{c}_{new} = \underline{c} - 1, \bar{c}_{new} = \bar{c} - 1, \gamma_{new} = 0, \delta_{new} = \bar{c} + \delta - \underline{c} - 1)_{LR}$$

Else

$$\tilde{c}_{new} = (\underline{c}_{new} = \underline{c} - 1, \bar{c}_{new} = \bar{c} - 1, \gamma_{new} = \gamma - 1, \delta_{new} = \bar{c} + \delta - \underline{c} - 1)_{LR}$$

End if

Compute PSE ($\tilde{c}_{new}, \tilde{c}$)

While (PSE > Risk) do

While (PSE > Risk) and ($\delta_{new} > 0$) do

$$\tilde{c}_{new} = (\underline{c}_{new} = \underline{c}_{new}, \bar{c}_{new} = \bar{c}_{new}, \gamma_{new} = \gamma_{new}, \delta_{new} = \delta_{new} - 1)_{LR}$$

Compute PSE ($\tilde{c}_{new}, \tilde{c}$)

Wend

If $\underline{c}_{new} - 1 < 0$ then it's impossible to find the flow with a total risk coefficient < Risk, Exit

End if

If $\gamma_{new} = 0$ then

$$\tilde{c}_{new} = (\underline{c}_{new} = \underline{c}_{new} - 1, \bar{c}_{new} = \bar{c}_{new} - 1, \gamma_{new} = 0, \delta_{new} = \bar{c} + \delta - \underline{c}_{new})_{LR}$$

Else

$$\tilde{c}_{new} = (\underline{c}_{new} = \underline{c}_{new} - 1, \bar{c}_{new} = \bar{c}_{new} - 1, \gamma_{new} = \gamma_{new} - 1, \delta_{new} = \bar{c} + \delta - \underline{c}_{new})_{LR}$$

End if

Compute PSE ($\tilde{c}_{new}, \tilde{c}$)

Wend

Else it's impossible to find the flow with a total risk coefficient < Risk and exit

End if

Next arc

Apply a maximum network flow algorithm with fuzzy input data for the network with new capacities.

End algorithm

3. Experiments and Results Analysis

Let us analyze the fuzzy network representing by the following table (Tab.1)

	Sa	Sc	ab	ac	ad
\tilde{f}	(1,3,1,1)	(3,7,2,1)	(0)	(0)	(1,3,1,1)
\tilde{c}	(1,3,1,1)	(3,7,1,1)	(4,5,1,1)	(5,6,1,1)	(1,3,1,1)
	bT	cb	cd	db	dT
\tilde{f}	(2,7,1,1)	(0)	(3,7,2,1)	(2,7,1,1)	(2,3,2,1)
\tilde{c}	(2,7,1,1)	(1,4,1,1)	(4,7,2,3)	(3,7,1,1)	(2,3,2,4)

Table 1. First illustrative network with fuzzy flows and fuzzy capacities

After the PSE coefficient calculating for every arc in this network we'll get the following results (Tab.2):

	Sa	Sc	ab	ac	ad
PSE	1	1	0	0	1
	bT	cb	cd	db	dT
PSE	1	0	1	1	1

Table 2. PSE calculating.

On the basis of this result it is possible to calculate the total network risk coefficient $RISK = 1$.

To illustrate the maximum network flow algorithm with fuzzy capacities and predetermined risk let us consider a network G with following parameters (Tab.3):

	Sa	Sb	AT	bT
\tilde{c}	(4,5,1,1)	(4,5,2,2)	(5,6,3,1)	(3,4,2,1)

Table 3. Second illustrative network with fuzzy capacities

Our task is to find a maximum flow in this network where the total risk of inconsistency situation appearance should be lower than 50%. On the first step we should define a new capacities for each arc in the network as it shown on figure.2.a-d.

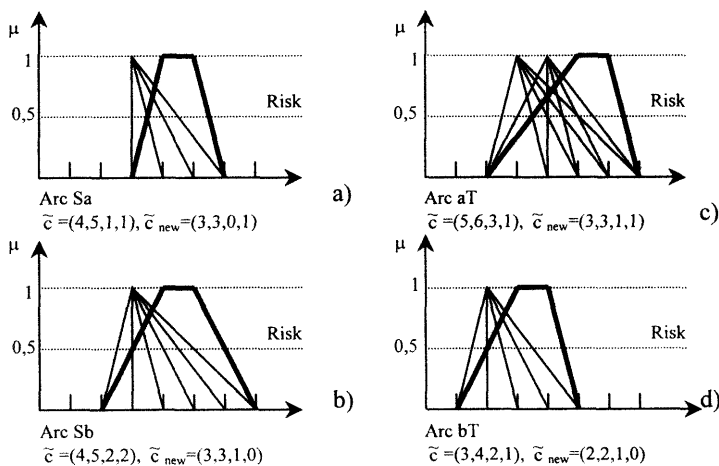


Figure 2. Risk constraint capacities defining.

And on the second step it is necessary to define a maximum fuzzy flow in the network with new capacities. After the applying of the max flow algorithm with fuzzy input data we'll get the following result: $\tilde{f}(Sa) = (3,3,1,1)$, $\tilde{f}(Sb) = (2,2,1,0)$, $\tilde{f}(aT) = (3,3,1,1)$, $\tilde{f}(bT) = (2,2,1,0)$.

After the computing of the total risk coefficient in the network shown in the table 1 we can in general estimate this final result as following: it is possible to pass through the network from 1 to 12 product units and most likely from 4 to 10 units. But in this case the possibility

of the inconsistency situation appearance will be equal to 1. The second illustrative example concerning the max flow finding algorithm with predetermined risk level we can estimate as following: it is possible to pass through the network from 3 to 6 product units and most likely 5 units. And in this case the possibility of the inconsistency situation appearance will be less then 50%. These approaches should help the decision-maker to build more informative network flow models and to control the failure situations in the fuzzy environment.

4. Conclusions

During the analysis of fuzzy networks it is not always possible to find the biggest fuzzy number because they can be covered with considerable proportions and the risk of inconsistency situation appearance may take place. In this situation a decision-maker have to find a compromise between max flow value and failure possibility value. The first method of finding the risk level value in fuzzy network is based on one of the integrated superiority coefficients PSE which is used in fuzzy numbers comparison operations. The second method of finding the fuzzy maximum flow in the network with predetermined risk value is based on the step-by-step decreasing of the capacity values to satisfy the risk constraint. The proposed approaches allows the decision-maker to analyze more flexible the flow networks with fuzzy characteristics and to make a beforehand decision on flow control to prevent network blocking.

References

- [1] M.Blue, B.Bush, J.Puckett, Applications of fuzzy logic to graph theory, Technical report in Los Alamos national laboratory LA-UR-96-4792, 1996
- [2] H.S. Shih, E.S. Lee, Fuzzy multi-level minimum cost flow problems, *Fuzzy sets and systems*, 107 (1999).
- [3] P.Diamond, A fuzzy max-flow min-cut theorem, *Fuzzy sets and systems*, 119 (2001).
- [4] Fundamentals of fuzzy sets, ed. by D.Dubois and H.Prade, 2000.
- [5] D. Dubois, H. Prade, Applications a la representation des connaissances en informatique, Masson, 1988.
- [6] R.Tyshchuk, Maximum network flow algorithm with fuzzy input data, Proceedings of the 5th International Conference on Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies, Japan, 2001
- [7] L.A. Zadeh, Fuzzy Sets, *Inform. and control*, 8, 1965.

Fuzzy Model of Student's Behaviour Generated by Means of GMDH Algorithms

Pavel NÁPLAVA and Miroslav ŠNOREK

*Department of Computer Science and Engineering, CTU FEE Prague
Karlovo náměstí 13, Prague, Czech Republic
naplava@cslab.felk.cvut.cz, snorek@cslab.felk.cvut.cz*

Abstract. This paper presents a method of modelling based on a data set by means of the GMDH Fuzzy Rule Induction algorithms. The data set contains information about fresh students of the Faculty of Electrical Engineering, Czech Technical University of Prague (CTU, FEE). They reflect student's personality, type and results from the high school and entrance examination results and serve as input vectors for prediction of his/her study results after the 1st semester.

Keywords: *Group Method of Data Handling (GMDH), Parametric Algorithms, Non Parametric Algorithms, Mean Absolute Percentage Error(MAPE)*

1. Introduction

The motivation for using the GMDH neural network is that its structure is created during the learning phase (we do not have to think about it before) and they are able to select the important input variables themselves.

2. Description of GMDH Fuzzy Rule Induction

GMDH was introduced in 1966 by A. G. Ivakhnenko [1]. It is based on an inductive learning algorithm for complex system modelling. It generates many alternate model variants (neural networks) with growing complexity by inheritance, mutation and selection until the best fitting one is found. In case when modelling objects with fuzzy characteristics (our case), GMDH nonparametric algorithms representing models of intervals of data or clusters are used. One of the most used nonparametric algorithms is Fuzzy Rule Induction (FUZZY) algorithm.

Algorithm generates a set of AND rules in layers and selects only the best ones until the error of models decreases. In the end it creates OR combination of the best selected AND rules from the last layer.

3. Experiments and their results

We experimented with data sets from years 1998 and 2000. For each data set we constructed more than one model with respect to different types of high schools. An example of the fuzzy model quality is illustrated in Fig. 1 (year 2000). Position of points shows divergencies between the estimated values of the model and the actual values (diagonal represents optimum).

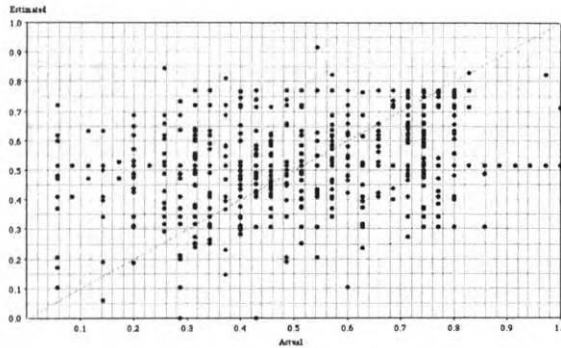


Figure 1. Quality of a Model

Qualities of all constructed models were measured with the Mean Absolute Percentage Errors. We found that it depends on the number of fuzzy sets. For smaller number of sets we obtained the value 15 %. If we add one more set the value raises to 17 %.

Our experiments also showed which parameters from input data set are the most relevant. They are parameters with the similar characteristic as the characteristic of the study at the University (entrance exams, state exam). Other parameters like age or results from mathematics and physics are not so important.

If we compare results obtained from FUZZY with the results obtained from parametric GMDH networks [2] we can say that the fuzzy models are able to cover the output values domain better than parametric algorithms. Parametric algorithms are able to generate only a short part of output value domain.

Next experiments are focused on creating models on pre-processed input data sets with another type of input values coding. We also want to create model for the next semesters because there are many “unstable” students in the 1st semester.

References

- [1] J. A. Müller and F. Lemke (2000) *Self-Organising Data Mining*, BoD Libri Hamburg, 225 pp.
- [2] P. Náplava and M. Šnorek (2001) *Modelling of Student's Quality by Means of GMDH Algorithms*, Proceeding of ESM'2001 Conference, Prague, Czech Republic, pp. 696–700, ISBN 1-56555-225-3.

Validation sets in R_{Δ} -fuzzy logics and RS-fuzzy logics

Rostislav Horčík

Center for Machine Perception, Department of Cybernetics,
Faculty of Electrical Engineering, Czech Technical University,
Technická 2, CZ-166 27 Praha, Czech Republic

Abstract. The validation set of a formula in a fuzzy logic is the set of all truth values which this formula may achieve. We extend recent results on characterizations of validation sets to R_{Δ} -fuzzy logics and RS-fuzzy logics.

Keywords: Fuzzy logic, Many-valued logic, Evaluation, validation set, Lukasiewicz logic, Gödel logic, Product logic

One way of representing vagueness is an enlargement of the set of truth values from $\{0, 1\}$ to $[0, 1]$. This leads to fuzzy logics [2, 4, 6]. These logics allow to achieve more degrees of satisfaction of a formula. In the classical logic, the formula $\varphi = \neg p \wedge p$ is always evaluated by 0. The same formula in a fuzzy logic, where the negation and conjunction are interpreted by the standard fuzzy negation and the minimum, may be evaluated by values greater than 0, but always at most $1/2$. More exactly, depending on the evaluation of p , the evaluation of φ may be any number from the interval $[0, 1/2]$. We express this fact by saying that $[0, 1/2]$ is the *validation set* of formula φ . In this paper we study the question of which validation sets may occur in various fuzzy logics.

The set of *truth values* used here is $[0, 1]$ and all fuzzy logics contain at least a conjunction interpreted by a continuous *t-norm*. Two basic t-norms are the minimum, T_G , and the product, T_P (see [2, 4, 6] for details).

A *residuum-based propositional fuzzy logic* (*R-fuzzy logic*) is a fuzzy logic in which the basic connectives are 0 (*false statement*), \wedge (*conjunction*) and \rightarrow (*implication*) with respective interpretations 0, T , R_T , where T is a t-norm and R_T is the corresponding residuum. This approach is described in [4]. The R-fuzzy logics corresponding to the basic t-norms T_G and T_P are *Gödel R-fuzzy logic* and *product R-fuzzy logic*.

Here we study which validation sets may occur in R_{Δ} -fuzzy logics and RS-fuzzy logics. The *validation set* of a formula φ is defined as

$$V(\varphi) = \{t(\varphi) \mid t \text{ is an evaluation}\}.$$

Results on validation sets in R-fuzzy and S-fuzzy logics can be found in [2, 5, 6].

A *residuum-based propositional fuzzy logic with Δ* (*R_{Δ} -fuzzy logic*) is an R-fuzzy logic in which the set of basic connectives is extended by a unary connective Δ with interpretation $t(\Delta\varphi) = 1$ iff $t(\varphi) = 1$, $t(\Delta\varphi) = 0$ otherwise. (This extension of R-fuzzy logics was introduced in [1].)

Theorem 1. *The validation sets in Gödel and product R_{Δ} -fuzzy logic are of one of the following forms: $\{0\}$, $\{1\}$, $\{0, 1\}$, $]0, 1[$, $[0, 1[$, $]0, 1]$.*

In R-fuzzy logics in which conjunction is not interpreted by a nilpotent t-norm, we have no disjunction dual to the conjunction. This has led recently to the following notion (see [3]): A *residuum-based propositional fuzzy logic with an involutive negation* (RS-fuzzy logic) is an R-fuzzy logic in which the set of basic connectives is extended by a unary connective \neg (negation) with interpretation $t(\neg\varphi) = 1 - t(\varphi)$.

Theorem 2. *A subset $V \subseteq [0, 1]$ is a validation set of some formula in Gödel RS-fuzzy logic iff $V \cap \{0, 1\} \neq \emptyset$ and $V \cap]0, 1[$ is an arbitrary union of the sets $]0, 1/2[$, $\{1/2\}$, $]1/2, 1[$.*

Theorem 3. *The validation sets in product RS-fuzzy logic are of one of the following forms:*

$$\{0\} \cup \bigcup_{i=1}^n I_i, \quad \{1\} \cup \bigcup_{i=1}^n I_i,$$

where $n \in \mathbb{N}$ and $I_i \subseteq [0, 1]$, $i = 1, \dots, n$, are intervals (open, closed or half-closed). The possible bounds of I_i form a countable dense subset of $[0, 1]$.

We studied two classes of fuzzy logics: R_{Δ} -fuzzy logics and RS-fuzzy logics. We gave a characterization of validation sets of formulas. This gives us information for comparison of semantical richness of these logics and their ability to describe vagueness. Among other results, we observe that inclusion of an involutive negation increases substantially possible degrees of partial satisfaction of formulas.

References

- [1] Baaz, M. (1996): Infinite-valued Gödel logic with 0-1 projector and relativisations. *Gödel'96: Logical foundations of mathematics, computer science and physics*, Lecture Notes in Logic 6, 23-33, Springer-Verlag.
- [2] Butnariu, D., Klement, E. P., Zafrany, S. (1995): On triangular norm-based propositional fuzzy logics. *Fuzzy Sets and Systems*, 69:241–255.
- [3] Esteve, F., Godo, L., Hájek, P., Navara, M. (2000): Residuated fuzzy logics with an involutive negation. *Arch. Math. Logic*, 39:103–124.
- [4] Hájek, P. (1998): *Metamathematics of Fuzzy Logic*. Trends in Logic, Volume 4. Kluwer, Dordrecht.
- [5] Horčík, R., Navara, M. (2001): Validation sets in fuzzy logics. In: M. Komorníková and R. Mesiar (eds.) *Proc. Int. Conf. Uncertainty Modeling '2001*, Bratislava, Slovakia, 82–90.
- [6] Klement, E. P., Navara, M. (1999): A survey on different triangular norm-based fuzzy logics. *Fuzzy Sets and Systems*, 101:241–251.

III. Evolutionary Computations and Miscellaneous Biologically Inspired Systems

This page intentionally left blank

Evolving Connectionist Systems for Modelling and Implementation of Adaptive, Intelligent Interactive Systems

Nikola KASABOV

*Department of Information Science,
University of Otago, P.O.Box 56, Dunedin, New Zealand,
phone: +64 3 4798319; fax: +64 3 4798311
nkasabov@otago.ac.nz¹*

Abstract. The paper presents the use of the evolving connectionist systems paradigm (ECOS) for the modelling and the implementation of adaptive systems for speech and image perception and recognition. ECOS are connectionist based systems that evolve their structure and optimise their parameters through fast, on-line learning from a dynamically changing stream of incoming data that does not have a pre-defined and known distribution. ECOS combine developmental learning of an individual system, optimisation through statistical criteria or evolutionary strategy, and evolutionary development through generations of populations of systems. These features are important for building adaptive speech and image recognition systems.

Keywords: *evolving connectionist systems, adaptive learning, speech perception; speech recognition; image recognition; colour quantisation*

1. Introduction

The complexity and dynamics of real-world problems, such as adaptive speech recognition and language acquisition, adaptive image recognition, multi-modal information processing, and many more, require sophisticated methods and tools for building on-line, adaptive, intelligent systems. Such systems should be able to: (1) learn fast from a large amount of data (using fast training); (2) adapt incrementally in an on-line self-organising mode; (3) dynamically create new modules – have open structure; (4) memorise information that can be used later; (5) interact continuously with the environment in a “life-long” learning mode; (6) deal with knowledge (e.g. rules), as well as with data; (7) adequately represent in their structure space and time; (8) deal with multiple modalities.

Several methods and systems have been developed so far that meet some of the criteria above and that have influenced the development of the evolving connectionist systems (ECOS). And these are: self-organising systems [1]; methods for on-line learning in connectionist structures [2,3]; methods for local area learning [4]; Resource Allocating Network RAN [5]; growing gas [6]; topological neural networks (NN) [7]. ECOS and the evolving fuzzy neural networks (EFuNN) in particular have features of growing and of structural adaptation during their

¹ Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, Private bag 92006, Auckland, New Zealand, Nik.Kasabov@auct.ac.nz, from 22 June 2002

operation. In terms of on-line neuron allocation, the EFuNN model is similar to the RAN model. These models allocate a new neuron for a new input example (x, y) if the input vector x is distant in the input space from the already allocated neurons, and also the output error evaluation $y - y'$, where y' is the produced by the system output for the input vector x , is above another threshold. Otherwise, neurons will be adapted to the example (x, y) . In terms of adaptive optimisation of many individual linear units EFuNN is close to the Receptive Field Weight regression (RFWR) model by Schaal and Atkeson. EFuNNs have also similarities with the Fritzke's Growing Cell Structures and Growing Neural Gas models, and with other dynamic radial basis function networks (RBFN) in terms of separating the unsupervised learning, which is performed first, from the supervised learning, applied next, in a two-type structure.

2. Evolving Connectionist Systems (ECOS) – Major Principles

2.1 ECOS architecture

The ECOS paradigm was published first in 1998 [8] was further elaborated afterwards. It allows an intelligent system to be automatically created from incoming data. An ECOS consists of nodes (units) that perform pre-defined functions, and connections between them. The system has a minimal initial structure that includes preliminary input and output nodes and few preliminary connections. Data is allowed to flow into the system so that if an input data vector is associated with a desired output vector, the system stores this association into a new node and creates new connections. Nodes and connections are created automatically to reflect the data distribution. The system's structure is dynamically changing after each data item is introduced.

The number of the input and output variables in ECOS can vary during the learning process thus allowing for more (or less) input and output variables to be introduced at any stage of the learning process. Input and output variables can have 'missing values' at any time of the learning process. If there is no output vector associated with an input vector, the system produces its own output vector (its own solution). If the desired output vector became known afterwards, the system will adjust its structure to produce this output, or one close to it, next time the same input vector is presented. The system continuously and adaptively learns from data to associate inputs to outputs and to cluster the data through allocating nodes to represent exemplars of data.

The learning process in ECOS is achieved through interaction with the environment which supplies the data flow and reacts to the output produced by the system.

In addition, the system provides the knowledge it has learned in the form of fuzzy IF-THEN rules. The ECOS framework implies that a system evolves through its operation in an interactive way. The more data are presented to the system, the more the system evolves. The learning process is on-line, life-long.

Nodes and connections in an ECOS system can be created, modified, merged, and pruned, in a self-organising manner, similar to how the human brain learns through creating and wiring neuronal structures. The system's structure grows or shrinks depending on the incoming data distribution and on pre-set parameters. Through the process of evolving from data the system learns the rules of its own behaviour. The rules that constitute the system's knowledge can be reported and/or extracted at any time of the system operation.

2.2 EFuNNs

One realisation of ECOS is the evolving fuzzy neural network EFuNN [8-10]. EFuNNs are models for evolving supervised learning from data that have five-layer structure where nodes and connections are created/connected as data examples are presented. An optional short-term memory layer can be used through a feedback connection from the rule (or 'case') node layer. The input layer of an EFuNN represents input variables. The second layer of nodes (fuzzy input neurons, or fuzzy inputs) represents fuzzy quantisation of each input variable space. For example, two fuzzy input neurons can be used to represent "small" and "large" fuzzy values. Different membership functions (MF) can be attached to these neurons (triangular, Gaussian, etc.). The number and the type of MF can be dynamically modified in an EFuNN. New neurons can evolve in this layer if, for a given input vector, the corresponding variable value does not belong to any of the existing MF to a degree greater than a membership threshold. A new fuzzy input neuron, or an input neuron, can be created during the adaptation phase of an EFuNN. The task of the fuzzy input nodes is to transfer the input values into membership degrees to which they belong to the MF. The third layer contains rule (case) nodes that evolve through supervised or unsupervised learning. Examples of the latter are given in [10]. The rule nodes represent prototypes (exemplars, clusters) of input-output data associations, graphically represented as an association of hyper-spheres from the fuzzy input and fuzzy output spaces.

Each rule node r in an EFuNN is defined by two vectors of connection weights – $W1(r)$ and $W2(r)$, the latter being adjusted through supervised learning based on the output error, and the former being adjusted through unsupervised learning based on similarity measure within a local area of the problem space. The fourth layer of neurons represents fuzzy quantisation for the output variables, similar to the input fuzzy neurons representation. The fifth layer represents the real values for the output variables.

The EFuNN evolving algorithm includes also: aggregation of rule nodes (i.e. merging rule nodes); pruning of rule nodes and connections, and other operations.

In the aggregation procedure two thresholds are used $T1$ and $T2$, and every two rule nodes for which the distance in the input space is less than $T1$ and the distance in the output space is less than $T2$ get aggregated. A normalised distance as a measure for the similarity between two vectors (two rule nodes represented by their connection weight vectors $W1$ and $W2$) is used in such a manner that providing the two vectors have been normalised and all their values are inside the interval $[0,1]$, the distance is also inside this interval.

EFuNNs can operate in several modes depending on the number of the rule nodes for which activation is propagated to the output layer: one of n (1-of- n); many of n (m-of- n).

A version of EFuNN - dynamic EFuNN (DEFuNN) uses fuzzy rules of Takagi-Sugeno type, rather than the EFuNN Zadeh-Mamdany rules [10]. The unsupervised version of EFuNN does not use output nodes. Instead, it evolves rule nodes based on similarities between the already evolved nodes and new data. One version of such systems is the evolving SOM (ESOM) [10, 13].

Simulators of different types of EFuNNs are available from the WWW:

<http://divcom.otago.ac.nz/infosci/kel/CBIIS.html>.

2.3 ECOS for on-line learning and evolutionary optimisation

During the on-line learning ECOS should be optimised "on the fly". This can be done with the use of the following techniques:

(a) Statistical and information theory criteria – probability of data assignment to each rule node, that defines how stable this node will be when accommodating new data or when aggregation with other rule nodes is performed; entropy can be used as a global criterion

(b) Applying evolutionary strategy to optimise in an on-line mode all (or the main) parameters of ECOS – structure, receptive fields, error thresholds, connection weights, learning rate, etc. Simultaneously evolving of many ECOS in a population and regularly applying a genetic algorithm for selecting the best (or several good) ECOS to continue with. This is a combination of on-line learning and regular off-line optimisation, say every 1000 examples or 1000 secs [10].

All the above techniques have been applied on different applications.

3. ECOS for modelling the emergence of speech clusters

Spoken languages evolve in the human brain through incremental learning and this process can be modelled to a certain degree with the use of evolving connectionist systems. Several assumptions have been hypothesised and proven through simulation in this chapter:

- (a) the learning system evolves its own representation of spoken language categories (phonemes) in an unsupervised mode through adjusting its structure to continuously flowing examples of spoken words (a learner does not know in advance which phonemes are going to be in a language, nor, for any given word, how many phoneme segments it has);
- (b) learning words and phrases is associated with supervised presentation of meaning;
- (c) it is possible to build a 'life-long' learning system that acquires spoken languages in an effective way, possibly faster than humans, provided there are fast machines to implement the evolving, learning models.

This application is presented in [11, 10].

4. ECOS for Adaptive Speech Recognition

An adaptive speech system can learn spoken phonemes, words and phrases. New words, pronunciations, and languages can be introduced to the system in an incremental, adaptive way. Such method that utilises ECOS is presented in [10].

Here, an evolving system for spoken digits recognition is developed. In order to assess the performance of the evolved EFuNN in this application, a comparison with the Linear Vector Quantization (LVQ) method is presented. Clean training speech data is used to train both the LVQ and the EFuNN models. Noise is introduced to the clean speech test data to evaluate the behaviour of the recognition systems in a noisy environment. Two different experiments are conducted with the use of the standard EFuNN learning method from chapter 3. In the first instance, car noise is added to the clean speech. In the second instance office noise is introduced over the clean signal. In both cases, the signal to noise ratio SNR ranges from 0 dB to 18 dB.

The results for car noise are shown in Fig. 1. The word recognition rate (WRR) ranges from 86.87% at 18 dB to 83.33% at 0 dB. The EFuNN method outperforms the LVQ method, which achieves WRR=82.16% at 0 dB.

5. ECOS for Adaptive Image and Video Recognition

In on-line processing of image information it is assumed that a continuous stream of images or videos flows to the system and the system always adapts and improves its ability to

classify, to recognize, to identify new ones. There are many tasks in the image recognition area that require on-line, incremental learning. Some application oriented models and experimental results of using ECOS (ESOM, EFuNN), along with other specific or generic methods for image processing, are presented in [10,13]. Such tasks are: On-line colour quantisation; On-line image classification; On-line video-camera operation recognition.

6. ECOS for Multi-modal Systems

Many processes of perception and cognition are multi-modal, involving auditory-, visual-, tactile-, and other type of information processing. All these processes are extremely difficult to model without having a flexible, multi-modular evolving system in place. Some of these modalities are smoothly added at a later stage of the development of a system without the need to “reset” the whole system. Such a system is presented in [10].

7. Conclusion

ECOS combine both known AI features and new features that make them promising techniques for the future development of intelligent systems and information sciences especially in the area of adaptive signal processing, adaptive speech and image recognition, multi-modal processing.

Acknowledgement

The development of ECOS and their applications was supported through a grant UOOX0016 funded by the Foundation for Research, Science and Technology, and the New Economy Research Fund, New Zealand.

References

- [1] Kohonen, T., Self-Organizing Maps, second edition, Springer Verlag, 1997
- [2] Freeman, J., D. Saad, “On-line learning in radial basis function networks”, *Neural Computation* vol. 9, No.7 (1997).
- [3] Heskes, T.M., B. Kappen, “On-line learning processes in artificial neural networks”, in: *Math. foundations of neural networks*, Elsevier, Amsterdam, 199-233, (1993).
- [4] Moody, J., Darken, C., Fast learning in networks of locally-tuned processing units, *Neural Computation*, 1(2), 281-294 (1989)
- [5] Platt, J., “A resource allocating network for function interpolation, *Neural Computation*, 3, 213-225 (1991)
- [6] Fritzke, B. “A growing neural gas network learns topologies”, *Advances in Neural Information Processing Systems*, vol.7 (1995)
- [7] Gaussier, T., and S. Zrehen, “A topological neural map for on-line learning: Emergence of obstacle avoidance in a mobile robot”, In: *From Animals to Animats No.3*, 282-290, (1994).
- [8] Kasabov, N. Evolving fuzzy neural networks - algorithms, applications and biological motivation, in: T.Yamakawa and G.Matsumoto (eds) *Methodologies for the Conception, Design and Application of Soft Computing*, World Scientific, 1998, 271-274
- [9] Kasabov, N. Evolving fuzzy neural networks for supervised/unsupervised on-line knowledge-based learning, *IEEE Trans. On SMC, part B: Cybernetics*. Vol.31, No.6, December, 2001, 902-918.
- [10] Kasabov, N. *Evolving connectionist systems – methods and applications in bioinformatics, brain study and intelligent machines*, Springer Verlag, 2002, in print
- [11] Taylor, J., N.Kasabov, and R.Kilgour, *Modelling the Emergence of Speech Sound Categories in Evolving Connectionist Systems*, Proc. of the Joint Conference on Information Sciences, Atlantic City, 2000
- [12] Ghosh and Kundu (eds) *Soft computing and image processing*, Heidelberg, Physica-Verlag (Springer Verlag) (2000)
- [13] Deng, D. and N. Kasabov, On-line image recognition with evolving self-organised maps, *Neurocomputing*, (2002) in print

The Population Sizing Problem in (P)GAs: Experiments

Zdeněk Konfršt, Jiří Lažanský

*Czech Technical University, FEE, Department of Cybernetics,
Technická 2, Prague 6 16627, Czech Republic*

Abstract. In order to estimate the correct size of population, approximations of population sizing have been used in the area of genetic algorithms (GAs). They consider a function being approximated, a representation of individuals and a character of used operators. Computing environment of Matlab allows us, with help of the estimation model, to identify the right population size and to estimate the final overall quality of individuals. As far as the comparison of theoretical results with real runs of GAs is concerned, we have to face several factors, which influence mutual correspondence. The main factors such as used GA operators and values of GA parameters are addressed in the article. Major important issues are tested against the estimation model, some irregularities of experiments are discussed and clearly new ways of next research come up.

Keywords: *Genetic algorithms, Parameters of Gas, Population sizing model, Uniformly scaled functions*

1 Introduction

Holland [1,2] idealized the process in a GA¹ as a cluster of parallel and interconnected 2^k -armed bandit problems. The Holland's bandit problem was extended by De Jong [3] in the equation for population sizing. It represented the basic noise-to-signal equation. Although it was not extended in the De Jong's dissertation, it gave a first approximation on the population sizing problem.

In [4], the statistical decision theory was employed. They were modelling a GA run as competitions between the best and the second best BBs (building blocks). Those two BBs were represented with their mean fitnesses and fitness variances. One of the outputs was that the probability of the right choice in a single trail in a problem (with m equally sized partitions) is

$$p = \Phi\left(\frac{d}{\sqrt{2m'\sigma_{bb}}}\right). \quad (1)$$

where Φ is the cumulative distribution function (CDF), σ_{bb} is the average BB variance, m' is a number of competing partitions ($m - 1$) and d is the fitness difference between the best and the second best BBs.

¹In some cases [7], GAs and parallel GAs (PGAs) could be transformed between each other.

In [6], they used a well known one-dimensional random walk with absorbing barriers $x = 0$ and $x = n$ representing convergence to the wrong and the right solutions, respectively. The variables p and $1 - p$ are probabilities that the best BB takes over the second best or vice versa. They were using an initial seed defined as $x_0 = \frac{n}{2^k}$ (of k order) and presupposed several conditions. First, the competition takes place between the best and the second best BBs in a partition. Second, crossover and mutation do not destroy significant numbers of BBs. Third, boundaries of the random walk are absorbing. The well-known equation² [5] was employed to get the quality of a solution as the number of X partitions converged to the right BBs.

$$P_{bb} = \frac{1 - \left(\frac{1-p}{p}\right)^{n/2^k}}{1 - \left(\frac{1-p}{p}\right)^n}. \quad (2)$$

where P_{bb} is a percentage of well-converged partitions. The initial population is $x_0 = n/2^k$. The probability p is a probability that the best BB takes over the second best and it should hold $p > 1 - p$. The probability p must be obtained from the equation (1).

There, we sum up all the important ideas from the reviewed theory. The equation (2) gives us the quality of a final population based on the population size (n). The probability p of the equation (1) is a product of number competing "bits" (m'), a problem being optimized (σ_{bb}) and the fitness difference (d) according to the equation (1). Different population sizes (n) would reflect different proportions of BBs (P_{bb}). "Good" population sizes reflect proportions BBs close to "1" or just "1". The model does not concern a computing time but the population size. GA is done when the population has converged and there are no "big" changes in the population. So to obtain the right population size, we get the proportion of BBs equal to "1" and a test function being optimized is solved.

2 Domains of interest

The Gamler's ruin model was the model we worked with. Some tests had taken place against the model and they were published in [6]. Some tests were omitted with or without the reason. We prepared several tasks to understand the estimation model, to evaluate the utility and the usability of the model, and understand parameters and operators, required for performance reliability.

1. What are the characteristic of $p = \Phi(d, m')$ and the impact of k on $P_{bb}(n)$?
2. Investigate the types of crossover(1-p, 2-p, uniform) and crossover probability p_c , which match the model best.
3. Select types of test function being approximated and stopping criterion.
4. Compare various representations of individuals (int, char, bit).
5. Influence of the string size on $P_{bb}(n)$.
6. Compare $P_{bb_{theor}}$ and $P_{bb_{exper}}$ using mutation and crossover operators.
7. Is the use of mutation operator possible? And when?

²A result from the theory of random walk is that a particle will be captured by the absorbing barrier at $x = n$.

3 Developments

In this part, we concentrate on the tasks (1-2) from the list above. The tasks are dealt with one by one according to their enumeration. In Fig. 1 (left), the equation (1) is displayed based on the fitness difference d and the competing partitions m' as parameters in a 3-D graph. The value ranges are $d \in < 0.5; 3 >$, $m' \in < 2; 100 >$ and $\sigma_{bb}^2 = 0.25$. In Fig. 1 (middle and right), the above mentioned 3-D graph is sliced for either $d = \text{const}$ or $m' = \text{const}$ to get pleasant 2-D characteristics. In the middle graph, the slicing points are $m' \in \{5, 10, 20, 40, 80\}$, the steepest curve matches the smallest value m' . The slicing points are $d \in \{0.5, 1.0, 2.0, 3.0\}$ and the lowest curve corresponds to the smallest d in the right graph.

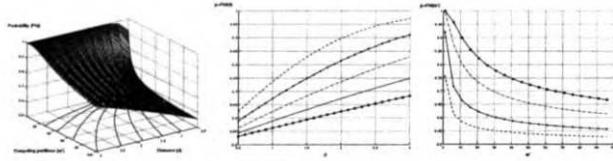


Figure 1: The probability of the right choice in a single trial $p = \Phi(d, m')$, $p = \Phi(d)$ and $p = \Phi(m')$

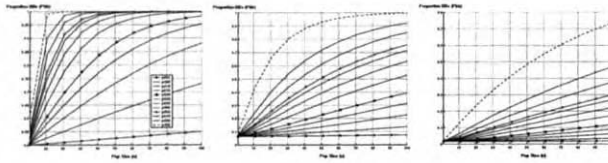


Figure 2: The proportion of BBs $P_{bb}(p, n, k)$ based on the variables n and k

In Fig. 2, the equation³(2) is shown for various probabilities p , $\sigma_{bb}^2 = 0.25$ and $k \in \{1, 4$ and $6\}$. The probabilities are marked in the graph according to a legend (e.g. the probability 0.501 as $p501$). The other images share the legend with the left one. The input values in the images differ just in k from the smallest (left) to the highest (right).

4 Experiments

Carried out experiments should answer usability of the estimation model and propose "appropriate" parameters for GAs. If not stated otherwise, a clean solid line is used for the model. The other line styles (solid, dashed, dotted, dashdotted with circles, pluses, stars and x-marks) represent data acquired from the experiments.

We started with testing three types of crossovers with different probabilities and their matching to the model. Then we examined the similarity between the model and real GA runs based on a string length. The set of graphs represents whether a string representation matters while using the model. Then we provided the characteristics based on the combination of mutation and crossover operators and their behaviour towards the model. In the end, we

³How to derive an average BBs variance σ_{bb}^2 based on a function being optimized was published in [4].

focused on the tuning of mutation probabilities. Some irregularities, inconsistencies and even errors were briefly tackled in Discussion.

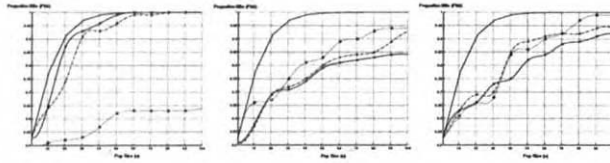


Figure 3: Crossovers-uniform (left), one-point (middle) and two-point (right)

Concerning GA parameters in experiments when we do not state any other parameters, we used uniform crossover with p_c probability, binary tournament selection $p_{ts} = 0.5$, generational replacement scheme, binary data type, the string length $m = 100$, mutation with probability p_m and the uniformly scaled test function—Onemax. The results are average of 10 independent runs with random initialization. We used two stopping conditions to avoid looping of GA. First stopping condition was "too similar" with the value 0.95. Second one was "maxiter" with the value 100. In case, that some of those parameters were diversified, the change is highlighted.

In Fig. 3, we present various type of crossovers with different probabilities with the output from the estimation model. Uniform crossover is represented in the left graph, the 1-p crossover is in the middle and the last one is the 2-p crossover. As it is clear from the graphs, the most successful is the uniform crossover operator with probability 0.5. It has the steepest increase of P_{bb} based on n and mirrors the estimation model. The uniform crossover with the probability 0.9 completely failed in this test domain. The other operators (1-p and 2-p crossovers) with various levels of probabilities (0.5, 0.7 and 1.0) does not work very well as figures show. In the next parts, we proceeded only with uniform crossovers.

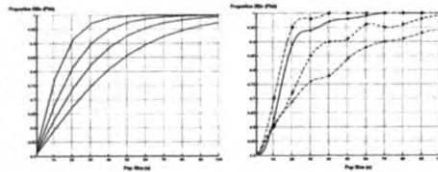


Figure 4: String size-theory (left) and experimental runs of GAs (right) based on $P_{bb}(n, m)$

In Fig. 4, we put together theoretical values received from the model and experimental values. The scope of this study is to get dependence of the population size on the string length. As it is presented, larger string sizes converge more slowly towards the proportion of BBs. The graphs with theoretical and experimental data approximately correspond.

In Fig. 5, we compared different string representations (bit, integer and char). We had in doubts whether a string representation would effect the model. It is also important to claim that larger alphabet representation means a longer time to convergence. The stopping condition "maxiter" was changed proportionally to the size of a character set. For a bit type, we employed the value 100. The integer representation used the value 500. And for a character

type, the stop value was adjusted to 1300. We presumed that a larger character set needs more time to converge because there are more patterns available. As it could be seen, the estimation model does not discern the data representation and confirms our assumption.

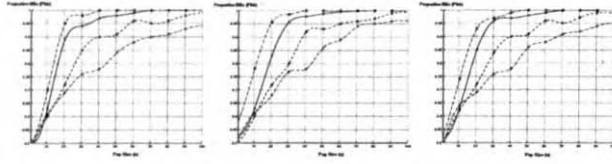


Figure 5: Data representation-bit (left), integer (middle) and char (right)

In Fig. 6 (left), the impact of mutation operator was studied. Two curves differ only in the size of p_c and the other two in p_m . In Fig. 6 (right), the main focus is on the levels of

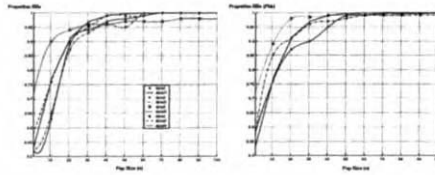


Figure 6: Experimental runs of GAs and their interpolations $P_{bb}(p_c, p_m)$ (left) and mutation tuning-experimental runs of GAs based on $P_{bb}(n, p_m)$ (right)

p_m . We tried to tune the mutation operator while the probability of crossover p_c was kept the same. Three levels (0.005, 0.001 and 0.01) (from left to right) of p_m were tested against the estimation model. They worked fairly well against the model. In the final stage, where the proportion of BBs is nearly "1", the disrupting effect of mutation is nicely visible. The mutation operator does not allow the population to converge completely.

5 Discussion

In Fig. 1, there is the 3-D graph when $\sigma_{bb}^2 = 0.25$. Decreasing of σ_{bb}^2 under 0.25, most of the graph gets saturated at $p = 1$. Increasing of σ_{bb}^2 over 0.25, it has the opposite effect. The graph gets flatter and the highest values of p will gradually decrease. Those approximations could be deduced from the equation (1). In a sense, the term σ_{bb}^2 could be called the parameter of "problem difficulty" because low σ_{bb}^2 will bring higher probability p and also higher P_{bb} .

As it has been shown in Fig. 2, the variable k is a considerable parameter because as it grows it quickly disables the ability of a GA to reach a sufficient proportion of BBs. In problems where the partition is created with more BBs ($k > 1$), higher probabilities of p are necessary for the estimation model.

In Fig. 3, the performance and reliability of crossovers were the main issue. We wanted to know if the model approximates all types of crossovers with all possible probabilities. As it is visible, it is not true. We can generalize that the 1-p crossover for the (uniformly) scaled function was the best choice. Those crossovers, which were not favourable, may need

a longer time to converge, higher population sizes for a steeper convergence and even different functions to optimize.

In Fig. 5, it is the best surprise that the model does not differ along the character set. However, the stopping condition must be regarded, it is a very promising result.

In Fig. 6 (right), as far as real runs of GAs and the model of GAs are concerned, they match fairly under one important condition. It is that zero or very low ($p_m < 0.001$) mutation is used. The estimation model was designed in a such way that the only disrupting process is a random generation of individuals in the beginning.

To create a broader and more complete study, we suppose that measurement of time to the complete convergence and applications to other optimization problems and real life problems should also be part of it. The type of selection scheme and selection probability p_s should also be put under investigation.

6 Conclusion

The fundamentals of the population sizing problem were explained and the relevant theory of population sizing was reviewed. The main stress was put on examining the population estimation model based on one-dimensional random walk. Some vital questions were raised. We warned of several "treacherous" aspects of the model. The carried out experiments gave appropriate and acceptable answers in most cases. Some relevant issues induced from experiments were discussed. From the study, we can generalize the following points:

- In scaled and uniformly scaled problems, the 1-p crossover gives a right mirror to the estimation model.
- The model scales very well according to a number of BBs (m).
- Type of string representation does not matter towards the model.
- Low mutation probabilities could be used without difficulty.

As it was stated in the beginning the findings could be applied not only to GAs but to PGAs of one population.

References

- [1] Holland J. H. (1973) Genetic algorithms and the optimal allocations of trials. *SIAM Journal of Computing* 2(2), 88–105
- [2] Holland J. H. (1975) *Adaptation in natural and artificial systems*. University of Michigan Press, Ann Arbor
- [3] De Jong K. A. (1975) *An analysis of the behaviour of a class of genetic adaptive systems*. Doctoral dissertation, University of Michigan, Ann Arbor
- [4] Goldberg D. E., Deb K. et al. (1992) Genetic algorithms, noise, and the sizing of populations. *Complex Systems* 6, 333–362
- [5] Feller W. (1966) *An introduction to probability theory and its applications*. 2nd ed., Volume 1, Wiley
- [6] Harik, G., Cantú-Paz E. et al. (1997) The gambler's ruin problem, genetic algorithms, and the sizing of populations. *Proc. of the IEEE International Conference on Evolutionary Computation*, 118–125
- [7] Konfršt Z. (2001) PGA and It's Behaviour. *Proc. of ICC'2001*, 399–404

A Constraint Handling Method for the Differential Evolution Algorithm

Jouni LAMPINEN

Lappeenranta University of Technology
Department of Information Technology
P.O.Box 20, FIN-53851 Lappeenranta, Finland
jlampine@lut.fi

Abstract. An extension for the Differential Evolution algorithm is proposed for handling nonlinear constraint functions. In comparison with the original algorithm, only the replacement criterion was modified for handling the constraints. Therefore the proposed approach is straightforward to implement and use. The user is not required to set any penalty parameters or any other additional search parameters. Furthermore, the user is not assumed to provide a feasible solution as a starting point for searching. In this article the proposed method is described and demonstrated by solving a suite of ten well-known test problems.

Keywords: evolutionary algorithms, differential evolution, global optimization, constraint functions, constraint handling

1. Introduction

A multi-constrained nonlinear optimization problem can be expressed as follows:

$$\begin{array}{ll} \text{Find} & X = \{x_1, \dots, x_D\}, \quad X \in \mathfrak{R}^D \\ \text{to minimize} & f(X) \\ \text{subject to constraints} & g_j(X) \leq 0, \quad j = 1, \dots, m \end{array} \quad (1)$$

A wide variety of evolutionary algorithm based approaches have been proposed for solving multi-constrained nonlinear problems [9]. However, handling multiple nonlinear constraint functions has been found rather difficult, and no totally satisfying method appears to be available, when considering all aspects involved.

In [9] Michalewicz and Schoenauer have surveyed evolutionary algorithms for constrained parameter optimization problems. They have classified the constraint handling methods applied with evolutionary algorithms into four categories:

- Methods based on preserving feasibility of solutions.
- Methods based on penalty functions.
- Methods which make a clear distinction between feasible and infeasible solutions.
- Other hybrids.

In general, the first two categories are undoubtedly the most widely applied with all types of nonlinear optimization algorithms, while the remaining two categories include a wide variety of less frequently applied approaches. The characteristic problems with the most frequently applied approaches are rather well known.

Methods *preserving feasibility of solutions* often require a feasible starting point for the search process. Unfortunately, a feasible starting point is often impossible to provide. The feasible region may be only a small subset of the whole search space. Finding a feasible

starting point randomly is often hopeless, or it takes much more time than the subsequent optimum-seeking process. Another drawback is that maintaining the feasibility often results in sticking into a suboptimal solution if the feasible solutions form a number of isolated islands around the search space due to multiple nonlinear constraints. This situation is typical for practical applications.

These problems can often be avoided by using *penalty function* approaches. However, typically penalty function methods require setting additional search parameters, like penalty parameters or weights for individual constraints. It is well known that these parameters affect the convergence performance as well as any other algorithm specific search parameter. Selecting parameter values can be complicated.

In particular, *underpenalization* of infeasible solutions (e.g. applying too low weight for a constraint) typically result in slow convergence towards feasible solutions, or in the worst case, no feasible solution will be found at all. *Overpenalization* typically results in a rapid convergence to a feasible solution, but often a premature convergence to suboptimal solution occurs simultaneously. If the feasible solutions form multiple disjointed regions around the search space, the population tends to converge to the first found island of feasible solutions, without exploring the others sufficiently.

Especially, if the number of constraints is relatively high, say >10 , finding the best penalty parameter values is a difficult optimization problem itself. Typically the user does not have enough problem specific information for selecting adequate settings *a priori*. So, the only alternative is to start with an educated guess, and then attempt to refine the settings by trial-and-error. This process is laborious and time consuming – it is rare to solve a nonlinearly constrained problem with a single run of the optimizer.

In real-life problems it is often impossible to judge *a priori*, whether there exist feasible solutions at all. Many penalty methods can provide a nearest feasible solution if the feasible set appears to be empty. After having the nearest feasible solution and the constraint function values at hand, it is often easy to judge which constraints are violated, why they are, and perhaps even reconsider the problem statements.

2. Differential Evolution Algorithm

The *Differential Evolution* (DE), introduced by Price and Storn [11, 12] can be classified as a *floating-point encoded* evolutionary optimization algorithm for global optimization over continuous spaces. The particular variant of DE used in this investigation was the *DE/rand/1/bin* scheme described in [11]. Since the original DE [12] does not include constraint handling, mostly various penalty methods have been applied, in the literature [7]. By applying penalty functions, a constrained problem can be effectively converted into an unconstrained one, which can then be solved with the DE. For example, the cost value, $f(X)$, to be minimized by DE can be composed by penalizing the objective function value, $f(X)$, with a weighted sum of constraint violations:

$$f'(X) = f(X) + \sum_{j=1}^m w_j \cdot \max(0, g_j(X)) \quad (2)$$

Typically a penalty function method will establish additional search parameters, like the weights, w_j , in Eq. 2. As mentioned, finding suitable parameter settings may be complicated. The need for improved constraint handling techniques is obvious.

3. New Approach for Constraint Handling

After a new trial solution has been evaluated with the objective function, the DE-algorithm applies a simple tournament selection rule for the decision, whether a trial solution vector, U , will be accepted into the population of the next generation, $G+1$, to replace the i :th member of the current population, the vector $X_{i,G}$:

$$X_{i,G+1} = \begin{cases} U & \text{if } f(U) \leq f(X_{i,G}) \\ X_{i,G} & \text{otherwise} \end{cases} \quad (3)$$

From the compared two vectors, U and $X_{i,G}$, the one providing lower objective function value will be selected to the next generation's population. The proposed modification for the DE's replacement rule is substituting the original replacement criteria with the following extended criteria, that is capable of handling also the constraints:

$$X_{i,G+1} = \begin{cases} U & \text{if } \left\{ \begin{array}{l} \left\{ \begin{array}{l} \forall j \in \{1, \dots, m\}: g_j(U) \leq 0 \wedge g_j(X_{i,G}) \leq 0 \\ \wedge \\ f(U) \leq f(X_{i,G}) \end{array} \right\} \\ \vee \\ \left\{ \begin{array}{l} \forall j \in \{1, \dots, m\}: g_j(U) \leq 0 \\ \wedge \\ \exists j \in \{1, \dots, m\}: g_j(X_{i,G}) > 0 \end{array} \right\} \\ \vee \\ \left\{ \begin{array}{l} \exists j \in \{1, \dots, m\}: g_j(U) > 0 \\ \wedge \\ \forall j \in \{1, \dots, m\}: \max(g_j(U), 0) \leq \max(g_j(X_{i,G}), 0) \end{array} \right\} \end{array} \right. \\ X_{i,G} & \text{otherwise} \end{cases} \quad (4)$$

Thus, when compared with the member of the current population, $X_{i,G}$, the trial vector U will be selected if:

it provides a lower or equal objective function value, while both the compared solutions are feasible, or

- it is feasible while $X_{i,G}$ is infeasible, or
- it is infeasible, but provides a lower or equal value for all constraint functions.

Still the objective is to select the better one out of the compared two solutions. This raises an obvious question; which one of the compared solutions is better when the constraint functions are included into comparison? Here it is considered that:

- If both the solutions are feasible, the one with lower objective value is better.
- Feasible solution is better than infeasible.
- If both compared solutions are infeasible, trial vector U can be considered less infeasible, and better than vector $X_{i,G}$, if it does not violate any of the constraints more than $X_{i,G}$ and if it violates less at least one of the constraints. This concept is analogous with the concept of Pareto-optimality (within constraint function space).

In addition, the trial vectors considered equally good as the compared population member are allowed to enter into the population in order to avoid stagnation of the algorithm [8]. Note, that in case of an infeasible solution, the replacement rule does not compare the objective function values at all. Therefore no unwanted selective pressure exists towards the search space regions providing low objective values, but infeasible solutions. This result in a fast convergence to the feasible regions of the search space, as demonstrated in the following.

4. Numerical Experiments

The proposed method was initially tested with a set of ten benchmark problems, often referred as *Michalewicz's problem set* [6, 9], despite the problems have been originally taken from [2, 3, 4]. The problems are summarized in Table 1, and the details are available via Internet: <http://www.lut.fi/~jlampine/testset.pdf>.

Problem	Number of Variables	Objective Function	Number of Constraint Functions ¹⁾			Problem no. in [6]
			Total	Linear	Nonlinear	
1	13	nonlinear	9	9	—	G1
2	7	nonlinear	4	—	4	G9
3	10	nonlinear	8	3	5	G7
4	8	linear	6	3	3	G10
5	2	nonlinear	2	—	2	G8
6	2	nonlinear	2	—	2	G6
7	5	nonlinear	6	—	6	G4
8	4	nonlinear	5 (8)	2 (2)	3 (6)	G5
9	2	nonlinear	1 (2)	—	1 (2)	G11
10	50	nonlinear	50 (100)	—	50 (100)	G3

¹⁾ Each equality constraint function was substituted with two inequality constraints requiring $-0.001 \geq g(X) \geq 0.001$. Value in brackets refers to the number of constraints after this conversion. Other constraints were required to be satisfied strictly.

Table 1. Summary of the test problems.

For each problem 1000 independent experiments were performed in order to study both the performance and the robustness of the method. For each problem, the worst, the average and the best objective function values, the number of objective function evaluations for obtaining the first feasible solution, and the CPU-time were recorded.

5. Results

In Table 3 the number of function evaluations for finding the first feasible solution were compared with the results obtained by pure random search. The results indicate that the proposed method introduces effectively a selective pressure towards the feasible region of the search space while no feasible solutions have been found yet.

Note from the Table 3, that the inverse $1/FE_{rs}$ of the average number of function evaluations for finding a feasible solution by random search estimates the size of the feasible region, F , in relation to the entire search space, S . The most difficult problems, 1, 3, 4, 6, 8 and 10, are all heavily constrained having ratio F/S less than 10^{-4} . For the problems 8 and 10, the random search did not find any feasible solution within reasonable time (more than a week of CPU-time used).

The results in Table 3 suggest that the proposed method is much more efficient in searching the first feasible solution than the random search. The difference was highest with the most difficult cases. For the problems 1, 3, 4, 6, 8 and 10, the proposed method found the first feasible solution 95% – 99% faster than the random search.

The justification for comparisons with the random search is that the random search is often applied for searching a feasible starting point for the methods requiring it. Also, it was relevant to find out, whether the proposed method is, in fact, any more efficient in finding the first feasible solution than the random search. Furthermore, the results of the random search make it possible to estimate the relative size of the feasible region, ratio F/S , for each test problem.

Problem	Mutation parameter, F	Crossover parameter, C_r	Population size, N_p	Number of generations	Number of function evaluations
1			20	4000	80000
2			25	1500	37500
3			35	5000	175000
4	$F = 0.9$	$C_r = 0.9$	30	9000	270000
5	for	for	20	500	10000
6	all	all	15	1000	15000
7	problems	problems	15	2000	30000
8			120	100000	12000000
9			30	1000	30000
10			40	200000	8000000

Table 2. The DE's control parameter settings applied. Note, that only the population size and the number of generations were coarsely varied to provide some problem specific adaptation.

Problem	Differential Evolution ¹⁾			Random Search ²⁾	Ratio F/S	Ratio FE_{avg}/FE_{rs}
	Worst FE_w	Average FE_{avg}	Best FE_b	Average FE_{rs}		
1	6409	3049	265	382946	2.61×10^{-4}	0.8%
2	464	103	1	194	5.12×10^{-3}	53%
3	23576	9809	69	881751	1.13×10^{-4}	1.1%
4	20589	8307	382	166479	6.01×10^{-4}	5.0%
5	375	84	1	112	8.93×10^{-3}	75%
6	645	350	12	15076	66.3×10^{-4}	2.3%
7	9	1.88	1	1.93	0.52	97%
8	11490917	2680623	1076210	160×10^{-9}	- ²⁾	0%
9	2672	525	1	983	1.02×10^{-3}	53%
10	228917	121561	77994	17.8×10^{-9}	- ²⁾	0%

¹⁾ Results are based on 1000 independent runs with both methods.

²⁾ No feasible solution found by the random search despite the reported high number of function evaluations (requiring more than a week of CPU time).

Table 3. Number of function evaluations for finding the first feasible solution.

Problem	Solution by	Number of Function Evaluations	Processor Time	Optimal or Best Known Solution $f(X^*)$	Found Solution			Error in Worst Case $f_w(X) - f(X^*)$	Average Error $f(X) - f(X^*)$
					Worst $f_w(X)$	Average $f(X)$	Best $f_b(X)$		
1	DE ¹⁾	80000	0.57 sec	-15.000 ²⁾	-15.000	-15.000	-15.000	0.000	0.000
	[10] ²⁾	n/a	n/a		n/c	-15.000	n/c	n/c	0.000
2	DE ¹⁾	37500	0.17 sec	680.630 ²⁾	680.630	680.630	680.630	0.000	0.000
	[10] ²⁾	n/a	n/a		n/c	680.718	n/c	n/c	0.088
3	DE ¹⁾	175000	1.08 sec	24.306 ²⁾	24.307	24.306	24.306	0.001	0.000
	[6] ²⁾	n/a	n/a		n/c	24.836	n/c	n/c	0.520
4	DE ¹⁾	270000	2.84 sec	7049.248 ²⁾	7049.248	7049.248	7049.248	0.000	0.000
	[6] ²⁾	n/a	n/a		n/c	7498.6	n/c	n/c	449.4
5	DE ¹⁾	10000	0.04 sec	0.095825	0.095825	0.095825	0.095825	0.000	0.000
	[6] ²⁾	n/a	n/a		n/c	0.095825	n/c	n/c	0.000
6	DE ¹⁾	15000	0.05 sec	-6961.814 ²⁾	-6961.814	-6961.814	-6961.814	0.000	0.000
	[6] ²⁾	n/a	n/a		n/c	-6948.1	n/c	n/c	13.7
7	DE ¹⁾	30000	0.22 sec	-31025.6 ²⁾	-31025.6	-31025.6	-31025.6	0.000	0.000
	[6] ²⁾	n/a	n/a		n/c	-30655.3	n/c	n/c	370.3
8	DE ¹⁾	1200000	68.5 sec	5126.484	5126.484	5126.484	5126.484	0.014	-0.014
	[5] ²⁾	n/a	n/a		n/c	5207.9604	n/c	n/c	81.46
9	DE ¹⁾	30000	0.07 sec	0.75000455	0.74900	0.74900	0.74900	0.001	-0.001
	[6] ²⁾	n/a	n/a		n/c	0.75	n/c	n/c	0.000
10	DE ¹⁾	800000	195 sec	-1.0000	-1.0252	-1.0252	-1.0252	0.0252	-0.0252
	[6] ²⁾	n/a	n/a		n/c	-0.9989	n/c	n/c	0.0011

¹⁾ Results are based on 1000 independent runs with DE.

²⁾ Results are based on only 10 independent runs of the compared algorithms.

³⁾ Reported by Floudas & Pardalos [2].

⁴⁾ Reported by Hock & Schittkowski [4].

⁵⁾ Found by DE in this investigation.

⁶⁾ Small equality constraint violation was allowed while converting them into two inequalities: $-0.001 \leq g(X) \leq 0.001$.

n/a - not available.

n/c - not comparable (different number of test runs performed).

Table 4. Summary of the experimental results and comparisons.

Table 4 summarizes the experimental results and compares the results with the best results reported in [5, 6, 10]. For all problems, the proposed method found the best known solution reported in the literature, except for the problems 4 and 7. For these problems even better solutions were found. All performed 1000 independent experiments for each problem returned practically identical results without failures.

The CPU-time for solving each test problems varied from a few seconds to a few minutes using a PC with Pentium III-500MHz processor. The CPU-times for the compared results were not reported in [5, 6, 10]. However, CPU-times ranging from 1 to 8 hours for solving the same problems with Genetic Algorithm and Sun Sparc/20 processor are reported in [1]. This information enables a rough comparison, since the performances of the compared processors are within the same order of magnitude.

6. Conclusions

In this article an extension of the DE-algorithm for handling multiple constraint functions was proposed and demonstrated with a well-known test problem suite. In comparison with the original DE-algorithm, only the replacement operation was modified for handling the constraints. Therefore the proposed approach is easy to implement as well as straightforward to use. User does not have to provide a feasible starting point which is required by many other methods, or set any penalty parameters, as in cases for most penalty function methods. From the user point of view, the proposed method allows solving multi-constrained problems virtually as easily as unconstrained ones.

The proposed method appears to provide some significant advantages when compared with the other contemporary constraint handling approaches. While the experiments have been performed with the DE-algorithm, the approach can also be applied with other algorithms, e.g. many Genetic Algorithms and Evolution Strategies. Thus the approach can be generalized further on for other global optimization algorithms.

With all test cases studied here, the proposed method demonstrated promising effectiveness, efficiency and robustness. Despite of that, due to limited test problem set, only drawing preliminary conclusions is considered justified. However, the current results certainly justify and motivate the further work with the proposed method.

References

- [1] Camponogara, E., Talukdar, S.N.: A Genetic Algorithm for Constrained and Multiobjective Optimization. In: Alander, J.T. (ed.). *Proc. of the 3rd Nordic Workshop on Genetic Algorithms and their Applications*, Helsinki, Finland, August 1997, pp. 49–61. FAIS (1997).
- [2] Floudas, C.A., Pardalos, P.M.: *A Collection of Test Problems for Constrained Global Optimization Algorithms*. Lecture Notes in Computer Science, vol. 455. Springer-Verlag (1987).
- [3] Himmelblau, D.: *Applied Nonlinear Programming*. McGraw-Hill (1992).
- [4] Hock, W., Schittkowski, K.: *Test Examples for Nonlinear Programming Codes*. Lecture Notes in Economics and Mathematical Systems, vol. 187. Springer-Verlag (1981).
- [5] Joines, J.A., Houck, C.R.: On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems. In: *Proceedings of the First IEEE Conf. on Evolutionary Computation*, 27–29. June 1994, vol. 2, pp. 579–584. ISBN 0-7803-1899-4.
- [6] Koziel, S., Michalewicz, Z.: Evolutionary Algorithms, Homomorphous Mappings, and Constrained Parameter Optimization. *Evolutionary Computation* 7(1):19–44, 1999.
- [7] Lampinen, J.: *A Bibliography of Differential Evolution Algorithm*. Tech. Report. Available via Internet: <http://www.lut.fi/~jlampine/debiblio.htm>. Cited 30 Nov. 2001.

- [8] Lampinen, J., Zelinka, I.: On Stagnation of the Differential Evolution Algorithm. In: Ošmera, P. (ed.). Proc. of MENDEL 2000, 6th Int. Conf. on Soft Computing, June 7.–9., 2000, Brno University of Technology, Brno, Czech Republic, pp. 76–83.
- [9] Michalewicz, Z., Shoenauer, M.: Evolutionary Algorithms for Constrained Parameter Optimization Problems. *Evolutionary Computation* 4(1):1-32, 1996.
- [10] Michalewicz, Z.: Genetic Algorithms, Numerical Optimization and Constraints. In: Proc. of the 6th Int. Conf. on Genetic Algorithms, Pittsburgh, July 15.–19., 1995, pp. 151-158.
- [11] Price, K.V.: An Introduction to Differential Evolution. In: Corne, D., Dorigo, M., Glover, F. (eds.). *New Ideas in Optimization*, pp. 79–108. McGraw-Hill, London (1999).
- [12] Storn, R., Price, K.V.: Differential Evolution – a Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. *Journal of Global Optimization* 11(4):341–359, 1997.

Competing Heuristics in Evolutionary Algorithms

Josef Tvrdík, Ladislav Mišík, Ivan Křivý

University of Ostrava,

701 03 Ostrava, Czech Republic

tvrdik@osu.cz, Ladislav.Misik@osu.cz, krivy@osu.cz

Abstract. The paper deals with a class of evolutionary algorithms (EAs) for the global optimization. Special attention is paid to the controlled random search (CRS). Generalization of the EA is proposed with several heuristics competing with each other when generating new trial points. The condition for asymptotic convergence of the algorithm are briefly discussed. Two instances of the EA with competing heuristics were implemented and the experimental results obtained on several test functions are presented.

Keywords: Global optimization, Evolutionary algorithms, Heuristics, Convergence, Self-adaptation of algorithm

1 Introduction

We deal with a class of evolutionary algorithms defined as follows. Let $D \subset \mathcal{R}^d$. Denote by \mathcal{L} the system of all Lebesgue measurable subsets of D and λ the Lebesgue measure on \mathcal{L} . Let f be a real Lebesgue measurable function defined on D . The number $\mu = \inf\{t; \lambda(f^{-1}(-\infty, t)) > 0\}$, where $f^{-1}(A) = \{x \in D; f(x) \in A\}$, is called the *essential minimum* of f . The task is to find an arbitrary good approximation of μ . Let p_0 be a probability measure on (D, \mathcal{L}) which is positive on each open subset of D . Let N be a positive integer called the *size of population* and D^N the set of all populations. Let π be a mapping defined on D^N assigning to each population \mathcal{P} the probability measure $\pi(\mathcal{P})$ on (D, \mathcal{L}) . Finally, let $\{m_k\}$ be a sequence of numbers from interval $\langle 0, 1 \rangle$ and let \mathcal{C} be a rule according to which some points in the old population are replaced by new ones. We consider a class of *evolutionary algorithms* described as follows:

- I : Generate an initial population $\mathcal{P}_0 = \{x_1, x_2, \dots, x_N\}$ chosen as an independent identically distributed sample according to the probability measure p_0 and set $k = 0$.
- C_k : Copy a portion of M best points of \mathcal{P}_k directly into the new population. Here the best points are those with the lowest values of f and M is an integer from $\{1, 2, \dots, N - 1\}$.
- S_k : Select a new point at random according to the probability measure $p_{k+1} = \pi(\mathcal{P}_k)$ and include it to the new population when fulfilling the condition \mathcal{C} . Repeat this procedure until the new population is complete.
- M_k : With the probability m_{k+1} replace a randomly chosen point by its mutation.

R: Set $k = k + 1$ and go back to C_k .

Mutation of a given point is a point in D generated according to specified rules. Starting from the results [4, 6] the sufficient condition for the algorithm convergence is as follows: For any measurable subset $S \subset D$ denote by $p_k(S)$ the probability that the algorithm produces in the k -th step a new trial point belonging to the set S . Then the algorithm is convergent provided that for every set S such that $\lambda(S) > 0$ we have $\prod_{k=1}^{\infty} (1 - p_k(S)) = 0$.

2 Evolutionary Algorithm with Competing Heuristics

We consider an evolutionary algorithm without explicit mutation:

procedure EA

generate \mathcal{P} (an old population of N points taken at random from D)

repeat

find the worst point in \mathcal{P} , \mathbf{x}_{worst} , with the highest value of f

copy M best points of \mathcal{P} into new population \mathcal{Q} , $1 \leq M < N$

repeat

repeat

generate a new trial point \mathbf{y} by some heuristics applied to \mathcal{P}

until $f(\mathbf{y}) < f(\mathbf{x}_{worst})$

insert the next trial point into \mathcal{Q}

until \mathcal{Q} is completed to N points

replace \mathcal{P} by \mathcal{Q}

until stopping condition is true

If we set the parameter $M = N - 1$ we get a generalization of the controlled random search described by Price [5]. The heuristics in this procedure is any non-deterministic rule which gives a new point $\mathbf{y} \in D$. There is no reason for using the only heuristics in a given evolutionary algorithm. The idea of using more alternating heuristics during an optimization process appeared a few years ago (see e.g. [11, 9]). Let us have h of heuristics at disposal. A new trial point can be generated by any of the heuristics. Each heuristics can be selected at random with probability q_i . When the probabilities q_i , $i = 1, 2, \dots, h$, are changing depending on the success of the i -th heuristics during the process, the EA procedure becomes an evolutionary algorithm with competing heuristics. Such algorithm has the feature of self-adaptation based on evolutionary ideas.

The heuristics is successful in the current step of evolutionary process when it generates such a trial point \mathbf{y} that $f(\mathbf{y}) < f(\mathbf{x}_{worst})$. When n_i is the current number of the i -th heuristics' successes, the probability q_i can be evaluated as

$$q_i = \frac{n_i + n_0}{\sum_{j=1}^h (n_j + n_0)}, \quad (1)$$

where $n_0 > 0$ is a constant. Setting $n_0 > 1$ prevents a dramatic change in q_i by one random successful use of the i -th heuristics. There is another way how to appreciate the success of a given heuristics. It can be evaluated by using the relative position of the trial point insertion into the new population \mathcal{Q} . Let us suppose that the new population \mathcal{Q} of actual size L (after inserting a new trial point), $L \in \langle M + 1, N \rangle$, is ordered in such a way that f_1, f_2, \dots, f_L is a nondecreasing sequence of function values. The new trial point generated by the i -th

heuristics is inserted into the new population \mathcal{Q} on the position $l \in \langle 1, L \rangle$. We can measure the success of the i -th heuristics by its weight w_i defined as $w_i = (L - l + 1)/L$, $w_i \in (0, 1)$, and probability q_i can be evaluated as

$$q_i = \frac{W_i + w_0}{\sum_{j=1}^h (W_j + w_0)}, \quad (2)$$

where $W_i = \sum w_i$ over a previous period of the process and $w_0 > 0$ is an input parameter which can be set to the mean value of w_i , i.e. $w_0 = 0.5$.

To avoid the degeneration of evolutionary process it is sometimes useful to reset the current values of q_i to their starting values. When any probability q_i decreases below a minority limit δ , i.e. when $q_i < \delta$, the values of n_i or w_i are reset to 0 what means that the probabilities are reset to their starting values, $q_i = 1/h$, and the number of the resets is increased. It can be easily seen that the EA with competing heuristics is convergent if at least one of the heuristics in use is convergent itself, i.e. when it is used alone. In practice this can be ensured by including the heuristics generating a new trial points \mathbf{y} uniformly distributed over D . Let us denote this heuristics by random search.

A natural question arises how to estimate the probability that the random search produces a proper trial point. Suppose that the random search is included in the set of heuristics in use with its serial number 1. Then $q_1(k)$ is the probability of using the random search in the k -th step of the EA algorithm and p is the probability that this heuristics produces a new trial point in a set $S \subset D$ with $\lambda(S) > 0$ during all the process. The probability $p_k(S)$ that the random search generates a new trial point in $S \subset D$ just in the k -th step is given by $p_k(S) = q_1(k)\lambda(S)/\lambda(D)$ and therefore,

$$1 - p = \prod_{k=1}^n \left(1 - q_1(k) \frac{\lambda(S)}{\lambda(D)} \right), \quad (3)$$

where n is the number of new trial points generated during all the process. Denote $\delta = \gamma/h$ for a suitable $\gamma \in (0, 1)$. From the nature of the random search and the above condition for calculation of the probability q_i it follows that the value of $q_1(k)$ will almost uniformly decrease from the initial value $1/h$ to the minimum possible value δ in every period between two successive resets. Thus, the average value \bar{q}_1 of $q_1(k)$ over all the process can be reasonably estimated by using the mean of both these limit values,

$$\bar{q}_1 = \frac{1}{2} \left(\delta + \frac{1}{h} \right) = \frac{\gamma + 1}{2h}.$$

The equation (3) can be rewritten in the form

$$1 - p \approx \left(1 - \frac{\gamma + 1}{2h} \frac{\lambda(S)}{\lambda(D)} \right)^n. \quad (4)$$

For very small values of $\lambda(S)/\lambda(D)$ (these sets S are of the major interest) the right side of the previous relation can be linearly approximated by using its differential and after a simple rearrangement we have

$$p \approx \frac{n}{2h} (1 + \gamma) \frac{\lambda(S)}{\lambda(D)}. \quad (5)$$

Thus we have the following approximate relation between p and $\lambda(S)$

$$\frac{n \lambda(S)}{2h \lambda(D)} < p < \frac{n \lambda(S)}{h \lambda(D)} . \quad (6)$$

Of course, with respect to using the linear approximation, the validity of this relation is restricted to such sets S that $\lambda(S)/\lambda(D)$ are sufficiently small.

3 Heuristics

Some examples of heuristics for generating a new trial point y can be found in [1, 2, 3, 7, 8]. The heuristics used in our implementation of the EA with competing heuristics are described below and the results of preliminary experiments as well as their parameter setting is given in Table 1.

Heuristics inspired by evolutionary strategy (ES) generate a new trial point y according to the following rule

$$y_i = x_i^{best} + Y, \quad i = 1, 2, \dots, d,$$

where x_i^{best} is i -th element of vector \mathbf{x}^{best} , i.e. the point with the lowest value of f in \mathcal{P} , and Y is random variable, $Y \sim N(0, \sigma_i^2)$. Instead of the classic evolutionary strategy, where the values of σ_i^2 are adapted within evolutionary process by the one-fifth rule [3], we derive the values of these parameters from the current population \mathcal{P} . We used two variants of the ES heuristics. One denoted as *esbest-pop* computes σ_i from the whole population as follows:

$$\sigma_i = c (\max_{\mathbf{x} \in \mathcal{P}} x_i - \min_{\mathbf{x} \in \mathcal{P}} x_i) + \varepsilon, \quad i = 1, 2, \dots, d,$$

where $c > 0$ is an input parameter and $\varepsilon > 0$ saves the value of σ_i positive (in our implementation $\varepsilon = 1 \times 10^{-4}$). In the heuristics denoted *esbest-2pts* the standard deviation σ_i is evaluated from two points \mathbf{r}, \mathbf{s} taken at random from the population \mathcal{P} (except \mathbf{x}^{best}) using the formula

$$\sigma_i = c |r_i - s_i| + \varepsilon, \quad i = 1, 2, \dots, d,$$

where the symbols have the same meaning as above.

Randomized reflection of simplex [2] of $d+1$ points chosen at random from \mathcal{P} is described by

$$\mathbf{y} = \mathbf{g} + U(\mathbf{g} - \mathbf{x}),$$

where \mathbf{x} is the simplex point with the highest f -value in the *refl-worst* heuristics or a randomly taken simplex point in the *refl-rand* heuristics, \mathbf{g} the centroid of the remaining simplex points and U random variable uniformly distributed on $\langle 0, \alpha \rangle$, α being an input parameter.

Two kinds of heuristics are based on differential evolution [7]. The heuristics denoted as *de-rand* generates a point \mathbf{u} using the relation

$$\mathbf{u} = \mathbf{r}_1 + F(\mathbf{r}_2 - \mathbf{r}_3),$$

$\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3$ being mutually different points taken from \mathcal{P} at random and $F > 0$ is an input parameter. The heuristic denoted as *de-best* is based on the modification of the point \mathbf{x}^{best} by the following way

$$\mathbf{u} = \mathbf{x}^{best} + F(\mathbf{r}_1 + \mathbf{r}_2 - \mathbf{r}_3 - \mathbf{r}_4),$$

where $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4$ are again mutually different points taken at random from \mathcal{P} (except \mathbf{x}^{best}) and F is an input parameter. The new trial vector \mathbf{y} is given by the crossover of vectors \mathbf{u} and \mathbf{x} chosen at random from \mathcal{P} (except $\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_3, \mathbf{r}_4$ and \mathbf{x}^{best}) according to the following rule

$$y_i = \begin{cases} u_i & \text{if } U < C \text{ or } i = j \\ x_i & \text{otherwise} \end{cases}, \quad i = 1, 2, \dots, d,$$

where $C \in \langle 0, 1 \rangle$ is an input parameter, U random variable uniformly distributed on $\langle 0, 1 \rangle$. To prevent the case with no change of \mathbf{x} (e.g. when $C = 0$) at least x_j is replaced by u_j , j is taken at random from $\langle 1, d \rangle$.

Any heuristics described above does not guarantee that a new point \mathbf{y} belongs to D . In case $\mathbf{y} \notin D$ so called perturbation is applied [3].

4 Experiments and Results

The EA with eleven competing heuristics (see Table 1) was implemented in Matlab, version 6. In this table the reliability R is the percentage of the successful runs stopping at a point very near to the global minimum and NE the corresponding average number of objective function evaluations. Our testbed contains the first two of five De Jong's functions (the second one is also known as Rosenbrock's saddle), Ackley's function and Griewangk's function (see [7]). Some of them present a serious problem for many optimization algorithms. For each task

Table 1: Reliability and convergence rate of the CRS with one heuristic

Heuristic	Parameters		De Jong 1 $d = 3$		Rosenbrock $d = 2$		Ackley $d = 2$	
	c, F, α	C	R	NE	R	NE	R	NE
esbest-2pts	1	-	100	585	100	3380	0	-
esbest-pop	0.2	-	100	457	100	2603	0	-
de-best	0.5	0.5	100	702	100	1060	93	752
de-best	0.9	0.5	100	1403	100	2586	96	1266
de-rand	0.5	0.5	96	879	39	1393	60	850
de-rand	0.9	0.5	100	1174	82	2049	75	1038
refl-rand	2	-	100	892	100	745	99	1392
refl-rand	6	-	100	2364	100	1565	99	3556
refl-worst	2	-	100	875	100	576	99	1228
refl-worst	6	-	100	2463	100	1430	98	3413
average			99.6	1179	92	1739	72	1687

100 independent runs were carried out. Two variants of the EA with competing heuristics were tested: *Competing1* using Eq. (1) and *Competing2* using Eq. (2) for the evaluation of probabilities q_i . The common input parameters were set as follows: $N = 10d$ and stopping condition in the form $f_{(N/2)} - f_{(1)} \leq 1 \times 10^{-7}$ ($f_{(1)} \leq f_{(2)} \leq \dots \leq f_{(N)}$). The search is considered successful when $f_{(1)} < 1 \times 10^{-3}$ for Ackley's function and $f_{(1)} < 1 \times 10^{-6}$ for others. The additional input parameters were $\delta = 0.02$, $n_0 = 5$, $w_0 = 0.5$ and $M = N - 1$. The CRS algorithm with alternating heuristics [10] (denoted *Alternating*) use the same heuristics excepting the random search.

The experimental results as well as the comparison with *Alternating* are given in the Table 2. The column p contains the p -values of one-way ANOVA tests (hypotheses that the means

Table 2: Comparison of EAs with competing and alternating heuristics

	Alternating			Competing1			Competing2			
Function	<i>d</i>	<i>R</i>	<i>NE</i>	<i>R</i>	<i>NE</i>		<i>R</i>	<i>NE</i>	<i>p</i>	
De Jong 1	3	100	858	100	800	*	100	768	*	0.000
Rosenbrock	2	100	1111	100	996	*	100	930	*	0.000
Ackley	2	95	1137	98	1052	*	96	1007	*	0.000
Ackley	10	99	11881	99	10975	*	100	10428	*	0.000
Griewangk	10	62	10131	77	* 10520		72	9462		0.009

of *NE* are the same for all the three algorithms). The symbols * behind the values denote the statistically significant differences from the corresponding values for *Alternating* at the significance level of 0.05. It is evident that the EAs with competing heuristics are at least as reliable as the *Alternating* one and in most cases the values of *NE* are significantly lower. The *NE*₁ values for getting the best point of the population close to the global minimum are approximately the same as for differential evolution [7], in the case of Griewangk's function even significantly lower (*NE*₁ = 12, 752 for the differential evolution and *NE*₁ = 8, 481 for the *Competing2*).

5 Conclusions

The EA with competing heuristics was proposed. Two instances of the algorithm were implemented and tested on five functions. The results were compared with those based on alternating heuristics [10]. The new algorithm was found more reliable and faster. The conditions for asymptotic convergence of the algorithm were discussed with respect to the implementation of the algorithm. The algorithm can be easily applied to searching for the constrained global minimum. The self-adaptation feature based on the heuristics' competition makes the setting of input parameters easier compared with other evolutionary algorithms.

References

- [1] Bäck, T.: *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, New York (1996)
- [2] Křivý, I., Tvrdík, J.: The Controlled Random Search Algorithm in Optimizing Regression Models. *Comput. Statist. and Data Anal.* **20** (1995) 229–234
- [3] Kvasnička, V., Pospíchal, J., Tiňo, P.: *Evolutionary Algorithms* (in Slovak). Slovak Technical University, Bratislava (2000)
- [4] Mišík, L.: On Convergence of a Class of Evolutionary Algorithms. In: Ošmera, P. (ed.): *MENDEL 2000, 6th International Conference on Soft Computing*. Technical University, Brno (2000) 97–100
- [5] Price, W.L.: A Controlled Random Search Procedure for Global Optimization. *Computer J.* **20** (1976) 367–370
- [6] Solis, F. J., Wets, R. J-B.: Minimization by Random Search Techniques. *Mathematics of Operations Research* **6** (1981) 19–30
- [7] Storn, R., Price, K.: Differential Evolution – a Simple and Efficient Heuristic for Global Optimization. *J. Global Optimization* **11** (1997) 341–359
- [8] Tvrdík, J., Křivý, I.: Simple Evolutionary Heuristics for Global Optimization. *Comput. Statist. and Data Anal.* **30** (1999) 345–352

- [9] Tvrđík, J., Křivý, I., Mišík, L.: Evolutionary Algorithm with Competing Heuristics. In: Ošmera, P. (ed.): MENDEL 2001, 7th International Conference on Soft Computing. Technical University, Brno (2001) 58–64
- [10] Tvrđík, J.: Controlled Random Search with Alternating Heuristics (in Czech). In: Proceedings of MATLAB 2001 Conference. HUMUSOFT, Prague (2001) 424–429
- [11] Weinberger, J.: Program Tools for Optimization and Identification of Simulation Models of Complex Systems (in Czech). Ph.D. Thesis. Charles University, Prague (1991)

Acknowledgement

This research was supported by the grant no. 402/00/1165 of the Czech Grant Agency and the institutional research grant no. CEZ: J09/98:179000002.

Evolvable Computational Machines: Formal Approach

Lukáš Sekanina

Faculty of Information Technology at Brno University of Technology
Address: Božetěchova 2, 612 66 Brno, Czech Republic; sekanina@fit.vutbr.cz

Abstract. The paper introduces an original formal definition of the evolvable computational machine. Mathematical properties as well as impact to application design are investigated. The proposed approach is demonstrated on an evolvable non-uniform cellular automaton for generation of sequences.

Keywords: *Evolutionary algorithm, Computational machine, Formal approach*

1 Introduction

Evolutionary algorithms (EA) are traditionally used for optimization, but many works were devoted to evolutionary design [1] in recent years. The paper deals with evolutionary design of *computational machines* (CM). There are some typical examples of CM evolved successfully in past years: cellular automata [2], Turing machines [3], Boolean circuits [4], finite state machines [5], or artificial neural networks [6]. Also specialized types of EA suitable for a given computational model have been developed: cartesian genetic programming for digital circuits [7], cellular programming for cellular automata [2] or genetic programming for evolution of computer programs [8].

Theoretical computer science operates with formally defined objects. Various CM were formally defined in history of the field (there is large overview in [9]). Some formalisms have been also introduced in the field of EA [10, 11]. The reasons for formal approaches are straightforward since: (i) problem specification is rigorous, (ii) a mathematical apparatus can be applied to investigate properties, and (iii) tools for automatic analysis, verification and design can be developed.

Taking in account all the reasons of the previous paragraph, we have tried to determine a minimal collection of mathematical objects to establish formal definition of an *evolvable computational machine* (ECM). The definition should allow us to investigate basic properties of ECM formally. Then we were interested in back-impact of such definition on the fields of theoretical computer science, evolutionary algorithms and practical implementations.

2 Formal definitions

2.1 Computational Machine: Non-Uniform Cellular Automaton

Every CM holds its formal definition and definition of its computation. As an example, the following definition (designed with inspiration in [2, 9]) of a non-uniform cellular automaton

is introduced.

Definition 1: One-dimensional binary **non-uniform cellular automaton** with the finite number of cells is 8-tuple $A = (d, Q, N, R, z, b_1, b_2, c_0)$, where: $d = 1$ is a dimension, $Q = \{0, 1\}$ is a binary set of states, N denotes a neighbourhood, z denotes the number of cells, b_1 and b_2 are boundary values, c_0 is an initial configuration, and a mapping $R : C \rightarrow (Q^N \rightarrow Q)$ assigns to each cell in $C = \{1, 2, \dots, z\}$ a *local transition function* $\delta_1, \dots, \delta_z$, where $\delta_i : Q^N \rightarrow Q$. \square

If only a single neighborhood $N = \{-1, 0, 1\}$ is considered, then *global transition function* $G : Q^C \rightarrow Q^C$ is defined as:

$$G(c(i)) = \begin{cases} \delta_i(c(i-1), c(i), c(i+1)) & \text{for } i = 2 \dots z-1, \\ \delta_1(b_1, c(1), c(2)) & \text{for } i = 1, \\ \delta_z(c(z-1), c(z), b_2) & \text{for } i = z, \end{cases}$$

where δ_i denotes the local transition function (rule) of the i -th cell defined by R . Cells are indexed from 1 to z .

G is used to define a sequence of configurations c_0, c_1, c_2, \dots such that $c_j = G(c_{j-1})$, for $j \geq 1$. This sequence represents a *computation* of A .

2.2 General Evolutionary Algorithm

Definition 2: (see [10]) A **general evolutionary algorithm** is defined as an 8-tuple $E = (I, \Phi, \Omega, \Psi, s, \iota, \mu, \lambda)$ where:

- (i) I is the space of individuals (a set of chromosomes),
- (ii) $\Phi : I \rightarrow \mathbb{R}$ denotes a fitness function assigning real values to individuals.
- (iii) μ is the number of parent individuals, while λ denotes the number of offspring individuals.
- (iv) $\Omega = \{\omega_{\Theta_1}, \dots, \omega_{\Theta_k} \mid \omega_{\Theta_i} : I^\lambda \rightarrow I^\lambda\} \cup \{\omega_{\Theta_0} : I^\mu \rightarrow I^\lambda\}$ is a set of probabilistic genetic operators ω_{Θ_i} each of which is controlled by specific parameters summarized in the sets $\Theta_i \subset \mathbb{R}$.
- (v) $s_{\Theta_s} : (I^\lambda \cup I^{\lambda+\mu}) \rightarrow I^\mu$ denotes the selection operator, which may change the number of individuals from λ or $\lambda + \mu$ to μ , where $\mu, \lambda \in \mathbb{N}$ and $\mu = \lambda$ is permitted. An additional set Θ_s of parameters may be used by the selection operator.
- (vi) $\Psi : I^\mu \rightarrow I^\mu$ is the generation transition function which describes the complete process of transforming a population P into subsequent one by applying genetic operators and selection:

$$\begin{aligned} \Psi &= s \circ \omega_{\Theta_{i_1}} \circ \dots \circ \omega_{\Theta_{i_j}} \circ \omega_{\Theta_0} \\ \Psi(P) &= s_{\Theta_s}(Q \cup \omega_{\Theta_{i_1}}(\dots(\omega_{\Theta_{i_j}}(\omega_{\Theta_0}(P)))\dots)) \end{aligned}$$

- (vii) The termination criterion is $\iota : I^\mu \rightarrow \{\text{true}, \text{false}\}$ ¹. \square

Definition 2 is based on high-level description where the population of individuals is manipulated by genetic operators and undergoes a fitness-based selection process. This is captured in the generation transition function Ψ , iterated application of which generates a

¹Here $\{i_1, \dots, i_j\} \subseteq \{1, \dots, k\}$, and $Q \in \{\emptyset, P\}$.

population sequence and leads to definitions of the *running time* and the *result of EA* (see appropriate definitions in [10]). Once the definition of the general EA and its computation is available, we can immediately establish formal definitions of genetic algorithm, evolutionary strategy or evolutionary programming (as seen in [10]). Because the space of individuals I can be arbitrarily complex, e.g. genetic programming may also be defined in terms of Definition 2.

Definition 2 deals with the space of individuals only. In many cases it is helpful to define search algorithms (i.e. also evolutionary algorithms) with respect to an abstract *representation* of the search space. Then the representation space defines a set of chromosomes which will be manipulated (by genetic operators) during search, and the *growth function* defines mapping between chromosomes and solutions [11].

3 Evolvable Computational Machines

To evolve a CM, we have to define which *part* of the CM will be under evolution because some parts of CM are usually required to be invariable (static). For instance, the number of inputs and outputs as well as required logical function of a combinational circuit is given by specification (i.e. is invariable), but its internal structure (i.e. a connection that implements the required function) has to be designed and so it can be the *subject of evolution*. Therefore, the subject of evolution has to be understood as knowledge available a priori.

We introduce two basic construction steps of formal definition of ECM. Let us consider non-uniform cellular automaton (CA) according to Definition 1 for demonstration of the concept:

(i) The subject of evolution (i.e. what is encoded in the chromosome) has to be chosen. Suppose that CA rules have to be evolved and the rest of CA is invariable. Then the growth function $g : I \rightarrow A'$ defines how to create a CA (i.e. a machine) from its genetic information (a chromosome). A' denotes a subset of all CA and determines which CA may be constructed using g .

(ii) A *machine fitness function* $f : A' \rightarrow \mathbb{R}$ has to be defined since behavior of a given CA must be evaluated in the process of fitness calculation. And because the fitness function is defined as $\Phi : I \rightarrow \mathbb{R}$, the fitness function is just the composition $f \circ g$.

The previous steps can be generalized for any CM. Let M denote a set of all CM. Then g will take the form $g : I \rightarrow M'$, where $M' \subset M$. And finally, general evolvable computational machine can be defined as follows:

Definition 3: An *evolvable computational machine* EM is a quadruple $EM = (M', E, g, f)$, where M' denotes a set of computational machines, whose part is the subject of evolution, performed by an evolutionary algorithm E . The growth function $g : I \rightarrow M'$ assigns to each individual x ($x \in I$) in E a machine in M' . Function $f : M' \rightarrow \mathbb{R}$ is a machine fitness function. Fitness function Φ in the E is a composition $\Phi = f \circ g$. \square

Computation of ECM is determined by definitions of computation of a given EA and CM.

Definition 3 deals with a *single* fitness function. However, EA working in time-dependent environments were also investigated [12]. In many applications of ECM (e.g. in the field of evolvable hardware [13]), the fitness function is changed dynamically to reflect environmental changes. To provide simple formal framework for the machine evolution in dynamic environments, the following definition, which is based on definition of a set of fitness functions (called environmental functions) and on a mechanism of transition between them is

given. (The set of all mappings from M' to \mathbb{R} will be denoted $\mathbb{R}^{M'}$).

Definition 4: Machine environment is a triplet $\Xi = (\Gamma, \varphi_0, \varepsilon)$ where $\Gamma \subseteq \mathbb{R}^{M'}$ denotes the set of *environmental functions* specifying quality of a machine in a given environment. $\varphi_0 \in \Gamma$ is an *initial environmental function*. $\varepsilon : \Gamma \rightarrow \Gamma$ denotes a relation on the set of environmental functions that determines successive environmental function. \square

4 Consequences of the proposed approach

(i) Binding the concepts of the ECM and the machine environment together, one can formally describe really evolvable system with dynamically changing requirements for evolutionary design.

Definition 5: Evolvable machine based application is a pair (EM, Ξ) where EM denotes an evolvable machine and Ξ is a machine environment. An initial environmental function φ_0 in the Ξ is matched with the machine fitness function f , i.e. $f = \varphi_0$. \square

(ii) From a practical viewpoint, Definition 5 implies the architecture of evolvable components [14]. The idea of the evolvable component is based on the following schema: If E , M' and g are designed in EM properly, then their definition (and mainly implementation) may be encapsulated and so reused in some *class* of applications. For instance, it was shown in [15] that various image filters can be successfully evolved at hardware level only by re-using a single component and redefinition of the fitness function.

(iii) The function g may be viewed as employing of a genotype-phenotype mapping. For example, in case of genetic algorithm and CA, a binary string represents a genotype while CA represents a phenotype. It is evident that different phenotypes may obtain the same fitness value. In general, the growth function has to be surjective. Moreover, if g is a bijective function then the representation is *faithful* and Lemma 1 holds:

Lemma 1: Φ is an equivalence relation on the set I , and its equivalence classes consist of genotypes whose phenotypes share the same fitness values. \square

In a very natural way, this result can be exploited for study of landscape neutrality [4] or in Surry's approach which demonstrates how to start from precise statements of beliefs about the relationship of problem structure to fitness and then mathematically derive representation along with appropriate operators in order to construct a problem-specific EA [11].

(iv) Assume that some Turing machine (TM) is evolved. So called halting problem of TM is undecidable [9]. It implies that an implementation of ECM must contain a mechanism to halt a candidate TM. It is also impossible to obtain a fitness value by an algorithm that analyses a candidate solution. Therefore, all candidate solutions must be executed to be evaluated.

(v) From a purely theoretical viewpoint, Definition 2 determines the fitness function as $\Phi : I \rightarrow \mathbb{R}$. But the requirement for \mathbb{R} as an infinite uncountable set is strong since a set of all TM is infinite but countable [9].

(vi) The proof of the following Lemma 2 is evident:

Lemma 2: If an environmental function φ_i is used to define the machine fitness function f at least two times, then the graph of the relation ϵ contains a loop. \square Open problems: (i)

Does any principal limitation (like No Free Lunch theorems [11]) determining the number of environmental functions which evolution is efficient for exist in the context of a given ECM?

(ii) Can the relation ε in Definition 4 be replaced by a function? Is it the case of real world applications of ECM? (iii) Are all the ECM isomorph in a sense? If so, software tool could be developed to support (semi)automatic design of ECM. (iv) Is it possible to consider evolvable component as a part of system theory?

5 An example: Evolvable Non-Uniform Cellular Automaton as a Generator of Sequences

This section demonstrates the proposed approach on the formal definition of an evolvable non-uniform CA with periodic boundary values (i.e. extreme cells are adjacent to each other), which is evolved to operate as a generator of the sequence: $seq = 5-6-7-8-9-10-11-12-13-14-15-0-1-2-3-4$. The CA is considered in the style of Definition 1. Simple genetic algorithm is derived from Definition 2 according to [10].

Evolvable non-uniform cellular automaton: $ECA = (A', GA, g, f)$

(i) **Cellular automaton** $A = (d, Q, N, R, z, b_1, b_2, c_0)$

$A' = \{(d, Q, N, R, z, b_1, b_2, c_0) \mid d = 1, Q = \{0, 1\}, N = \{-1, 0, 1\},$

$z = 4, b_1 = c(4), b_2 = c(1), c_0 = 0101\}$

R is the subject of evolution and R is represented as an 8bit table for each cell. The cardinality of A' is 2^{32} .

(ii) **Genetic algorithm** $GA = (I, \Phi, \Omega, \Psi, s, \iota, \mu, \lambda)$

$I = \{0, 1\}^{32}$ (i.e. an individual is a 32bit string – 8bits per cell)

$\Phi = f \circ g$

Ω contains two operators: mutation $m_{\{p_m=0.18 \text{ per bit}\}}$ and one-point crossover $r_{\{p_c=0.7\}}$ (see definition in [10])

s : 2-tournament selection with elitism (see definition in [10])

$\iota : t_{max} = 10^6$ (i.e. finish when the generation t_{max} is reached)

$\Psi = s \circ m \circ r$

$\lambda = \mu = 32$

(iii) **Growth function** $g : \{0, 1\}^{32} \rightarrow A', e(\vec{a}) = A = (d = 1, Q = \{0, 1\},$

$N = \{-1, 0, 1\}, R = \vec{a}, z = 4, b_1 = c(4), b_2 = c(1), c_0 = 0101)$ for $\vec{a} \in I$

(iv) **Machine fitness function** $f : A' \rightarrow \mathbb{R}$ and $f(e(\vec{a})) = \sum_{i=1}^{15} cmp(c_i, seq_i)$, where

$$cmp(c_i, seq_i) = \begin{cases} 1 & c_i = seq_i, \\ 0 & c_i \neq seq_i. \end{cases}$$

The parameters chosen in the example were set up after exhaustive experimental analysis (more than 25000 runs) where it was tested: population size (8-256), crossover probability (0-100%), mutation probability (1-10bits/chromosome) and selection mechanism (roulette wheel, 2-tournament, deterministic). Every run was repeated 100 times. The best solution (fitness = 10) appeared in 100 cases of 100 runs in generation 25505 on average. The evolution is very sensitive to change of parameters.

After the analysis of the whole state space of the problem (2^{32} states) using another program, it was recognized that the best fitness value 10 is shared by two solutions (CA rules 4E7A6633_h and 4A7A6633_h). It means there is not any CA that is able to approximate more than 10 numbers of the sequence seq . Thus all CA can be divided into 11 equivalence classes for this task.

For instance, if the sequence is $seq0 = 0 \dots 15$ then there are 9 equivalence classes or for the sequence $seq2 = 2, 3, \dots$ there are 10 equivalence classes. However, new machine fitness functions have to be designed for these sequences. All these machine fitness functions can be captured in Γ and provide *machine environment* according to Definition 4. Then *ECA* tries to adapt to changing requirements on production of sequences that are specified using the relation ϵ .

6 Conclusions

We have established formal definitions of ECM and machine environment to investigate basic theoretical properties of ECM formally. Definition 3 shows that general ECM can be defined independently of a given CM, EA and application and thus all ECM operates exactly in the same way.

The ideas of reusability and evolvable component are important from implementation viewpoint. Many problems have been introduced and these open problems will be the subject of future research.

Acknowledgment

The research was performed with the Grant Agency of the Czech Republic under No. 102/01/1531 *Formal approach in digital circuit diagnostic – testable design verification*. The author would like to thank Alexander Meduna for his fruitful comments.

References

- [1] Bentley, P. J. (1999) *Evolutionary Design by Computers*. Morgan Kaufmann Publisher
- [2] Sipper, M. (1997) *Evolution of Parallel Cellular Machines: The Cellular Programming Approach*. Springer-Verlag, Berlin
- [3] Vallejo, E., Ramos, F. (2001) *Evolving Turing Machines for Biosequence Recognition and Analysis*. In: Proc. of Genetic Programming European Conference EuroGP 2001, Springer-Verlag, 36–50
- [4] Miller J., Job, D., Vassilev, V. K. (2000) Principles in the evolutionary design of digital circuits – Part I and II. *Journal of Genetic Programming and Evolvable Machines*, Vol. 1(1–2,3), 8–35, 259–288
- [5] Sanchez, E., Pérez-Urbe, A., Mesot, B. (2001) Solving Partially Observable Problems by Evolution and Learning of Finite State Machines. In: *Evolvable Systems: From Biology to Hardware ICES 2001*, Springer-Verlag, 267–278
- [6] Yao, X. (1999) Evolving Artificial Neural Networks. *Proceedings of IEEE*, Vol. 87 (9), 1432–1447
- [7] Miller, J., Thomson, P. (2000) Cartesian Genetic Programming. In: Proc. of the Genetic Programming European Conference EuroGP 2000, LNCS 1802, Springer-Verlag, Berlin, 121–132
- [8] Koza, J. et al. (1999) *Genetic Programming III : Darwinian Invention and Problem Solving*. Morgan Kaufmann Publishers
- [9] Gruska, J. (1997) *Foundations of Computing*. International Thomson Publishing Computer Press
- [10] Bäck, T. (1996) *Evolutionary Algorithms in Theory and Practice*. Oxford University Press
- [11] Surry, P. D. (1998) *A Prescriptive Formalism for Constructing Domain-specific Evolutionary Algorithms*. PhD thesis, University of Edinburgh

- [12] Branke, J. (2001) Evolutionary Approaches to Dynamic Optimization Problems – Update Survey. In Proc. of the GECCO Workshop on Evolutionary Algorithms for Dynamic Optimization Problems, 27–30
- [13] Sanchez, E. et al. (1997) Phylogeny, Ontogeny, and Epigenesis: Three Sources of Biological Inspiration for Softening Hardware. In proc. of Evolvable Systems: From Biology to Hardware ICES96, Springer-Verlag, 35–54
- [14] Sekanina, L., Sillame, A. (2000) Toward Uniform Approach to Design of Evolvable Hardware Based Systems. In Proc. of Field Programmable Logic and Applications FPL2000, Springer-Verlag, 814–817
- [15] Sekanina, L. (2002) Image Filter Design with Evolvable Hardware. To appear in Proc. of the 4th European Workshop on Evolutionary Computation in Image Analysis and Signal Processing EvoIASP2002, Springer-Verlag, p. 12

Steady-State Evolutionary Algorithm with an Operator Family

L. GACÔGNE * **

* LIP6 - Université Paris VI 4 place Jussieu 75252 Paris 5°

tel : 01 44 27 88 07 fax : 01 44 27 70 00

** Institut d'Informatique d'Entreprise (CNAM) 18 allée J.Rostand 91025 Evry
Louis.Gacogne@lip6.fr

Abstract. A particular steady-state new strategy of evolution is studied in this paper. After comparison with GA and ES, we specially focus our attention on the choice of genetic operators, the way to apply them and finally how each generation is built from the previous one. Knowing that it is not possible to reach a universal heuristic able to choose the genetic operators and to manage them, we present a method where the genetic operators themselves are evaluated according to their performance. Thanks to this method, the improvement observed in order to optimize classical functions and the no relevant trials for adaptive rates, brings us to combine it with a very simple steady state algorithm with a small sized population, and we conclude by a recommendation about parameters like population size, updating and clearing rates.

Keywords: evolutionary algorithms - steady state genetic algorithms - adaptive operators

1. Introduction

Evolutionary computation comprises different techniques that have been inspired by biological mechanisms. Beyond the canonical genetic algorithm (GA) [Holland 75], [Goldberg 82, 89], many ways of research intend to accelerate evolution in view of optimization problems with a large space of candidate solutions. GA are inspired by the natural evolution rules where a population is updated, by selection and crossing-over, where mutations with very few probabilities may bring a modification for the individual behaviour in any direction, and then, may facilitate or not the reproduction ability. In standard GA, solutions are described in a binary encoding way, parents are picked according to the biased wheel principle and they are always replaced by their children. In evolution strategies (ES) [Schwefel 90], each parent may produce a number λ / μ of children (around 7) [Bäck 95, 97]. ES are a kind of accelerated evolution with the alternative to mix parents and children to keep the μ best in $ES(\mu + \lambda)$, or to remove the old generation in $ES(\mu, \lambda)$ by the new one. Another way is to keep the best between father and child (elitist Lamarck' evolution), in elitist-ESAO described below, or to update the worst part of old generation with the best part of the new one in SSGA. It is important to see that the best fitness and the average fitness are monotonic or not, according to those strategies [Bäck 95, 97].

For us, the hardest question raised by artificial evolution is to avoid a too homogeneous population. In order to study and compare original heuristics which intends to raise better results than classical GA or ES, we follow a previous work where we concluded that mixing different features with steady-state ESAO could give very fast results with respect to classical strategies. We present here tests about this last algorithm, may be not universal

[Wolpert, McReady 95], focusing on family of operators, updating rate, clearing and on the size of the population.

2. Algorithms with an operator family

The steady-state algorithm SSGA(μ , τ)

In the original version, [Whitley, Kauth 88], [Davis 91], [De Jong, Sarma 93], two parents are chosen according to the roulette wheel, crossover and mutations are applied and the two children replace the two worst individuals of the population. For all the following work, we shall call SSGA(μ , τ) the algorithm where each parent produces a child (with an operator among a family of predefined genetic operators) in such a way that μ parents have exactly an offspring of μ children, then, with a rate τ (for example 25%) the τ best children remove the τ worst parents.

The Evolutionary Strategy with adaptive operators: elitist-ESAO and steady-state-ESAO

A lot of papers introduce different kinds of specific genetic operators, following [Michalewicz 92] and [Tuson, Ross 96]. In our algorithm, we provide a set of various genetic operators moving during an evolution, according to their ability to decrease the fitness. We already use this algorithm for the tuning of fuzzy controller [Gacôgne 97, 99] where there were operators like "creation of rule", "suppression of a randomly chosen rule", "gaussian noise for one of the parameters" ... For this, a set OP of operators is available and updated as the population P is itself updated :

- 1) Let P_0 be random population of μ individuals, the chromosome encoding is defined according to the problem, it could be character string, symbols list, tree as in genetic programming, any structure mixing numeric and symbolic items.
- 2) Let OP_0 be another random list picked (with repetitions) among a list of genetic operators : mutations, transpositions for two genes, inversion of sequences, gaussian noise for numeric genes, migration, creation or suppression inside the chromosome, crossover over one or two sites, ... particular removing of a subtree by an other... All operators may be imagined according to the problem. Many particular works are looking for specific operators linked to a special representation [Kane, Schoenauer 97], [Jensen 92]. Each one will have a measure (0 at the beginning and cumulated along the evolution) for its ability to decrease the fitness f .
- 3) Each generation t , the sorted operators of OP_t are applied in the same rank to the individuals of the sorted population P_t . When one of them is applied to an element i , the op "score" is increased by $f(op(i)) - f(i)$, and in "elitist-ESAO" we remove at once the worst between i and $op(i)$. Thus, after sorting OP, the best of them will be applied to the best of P and so on.
- 4) If there is at least one progress during a generation, the best operator of OP is duplicated at the head and the last operator of OP is removed. By this way OP_{t+1} will change all the time during evolution.
Otherwise, when any amelioration is observed, the whole family of operators OP is reinitialized. Thus we avoid a convergence towards the same operator, which could have a role during a phase of evolution, but would exclude the others for the following evolution.
- 5) Evolution is stopped if the number of fitness evaluations reaches a threshold max, (or any other end condition) otherwise go back to step 3.

This algorithm will be called elitist-ESAO(μ) when the worst between father and son, is at once removed, and by steady-state-ESAO(μ , τ) we mix the two last ideas, updating P by removing the worst parents by the best children, so P is performed with a rate τ , but the i -th parent still produces in both cases the i -th child thanks to the i -th operator of OP.

Representation We encode all spaces like $[a, b]^{\dim}$, with solutions represented by a vector of decimal digits list in order to code real numbers in $[0, 1]^{\dim}$. For instance if $\dim = 1$, among $[7, 10]$, the integer list $[3;3;3;3;3;3;3;3;3;3]$ means $1/3$, and by dilatation to the interval $[7, 10]$, it will represent 8.

Operators We used the following operators to explore the neighbourhood (the "migration-0" is not only used to initialize the population, but also to replace double or similar individuals and moreover as an operator) :

migration-0: gives a completely new individual

migration-1: everything is replaced except the only first element of each component

migration-2: a random half of components is replaced

migration-3: a component is completely removed and replaced by a random one

mutation-1: mutation for a randomly chosen digit in a component

mutation-2: "small" mutation adding ± 1 to one digit.

addition: a new digit is added at the end of one component

transposition-1: two digits are swapped inside the same component

transposition-2: two components are swapped (for a dimension greater than 2)

copy: a component is removed by another duplicated (for a dimension greater than 2)

crossover-0: uniform crossover with a random other parent

crossover-1: one site crossover

crossover-2: two sites crossover

crossover-3: crossover only with one the 10% best individuals ("coral-fish" idea)

Adaptive updating rate Previous tests showed that the rate τ may be roughly between 25% and 50% with similar results. Empirically, when evolution stagnates, increasing the replacement rate often proves beneficial perturbation inside the population, in counterpart, as it brings in many new "bad" individuals, this entails a waste of computational efforts. So it is possible to argue this or the opposite and we tried the two ways. Then, for the first idea, let us assume $\text{amp}_0 = f_{\max} - f_{\min}$, the population initial amplitude. If the amplitude-ratio is close to 0 (too much concentration), we move the rate up toward the maximal rate t_{\max} , and if this ratio amp/amp_0 is close to the initial one (too much dispersion), then we move it down toward t_{\min} . Another basic idea is to make the same variations according to the position of the mean f_{avg} of the fitness distribution between f_{\min} and f_{\max} .

First idea, if $t_{\min} = 10\%$, the number of new children removing parents will be $\frac{\mu}{100} [90 - 80\min(1, \frac{f_{\max} - f_{\min}}{\text{amp}_0})]$.

For the second option, if $t_{\min} = 10\%$, this number will be $\frac{\mu}{100} [10 + 80\min(1, \frac{f_{\max} - f_{\min}}{\text{amp}_0})]$.

Choosing $t_{\max} = 1 - t_{\min}$, the third option is to set $t = t_{\min} + (1 - 2t_{\min}) \frac{f_{\max} - f_{\text{avg}}}{f_{\max} - f_{\min}}$.

As we can see later, any of those three options brings improvements.

Elimination (clearing) and adaptive elimination An elimination procedure may be also added in removing the worst of each pair of individuals, scrolling the population from the worst to the best after each generation. This elimination of neighboring chromosomes is performed only if some similarity degree is under a threshold, and then, a new random element can be introduced by migration to remove too similar chromosomes. Of course when this procedure is realized, it must be again followed by a population sorting.

So, too similar individuals may be removed in favor of the best of them. We compute the rate of similarity between two vectors in $[0, 1]^n$ as the minimum of the rates of common digits seen on the shortest length for each pair of components. It is a very simple fuzzy extension of equality adapted to the decimal representation in view of making a drastic elimination of similar individuals. For instance, with a function called *prox*, we could have $\text{prox}([1;2;3], [1;2;3;4;5;6]) = 100\%$ and $\text{prox}([1;2;3;4;5;6;7;8;9;0], [1;2;3;4;8;5;6;7;8;9;0]) = 40\%$.

3. Numeric tests

We first try different strategies for some frequently used test functions in various dimensions 2, 10 or more. All of them reach their minimum 0 and the bounds $[a, b]$ are not very relevant because they are strongly multimodal except the parabolic and DeJong convex functions. In all the following tests we note the average number of evaluations of the function reaches a threshold 10^{-6} except for Griewank function where we fix the threshold 10^{-4} .

De Jong function If $x = (x_1, x_2, \dots, x_{\text{dim}}) \in [-500, 500]^{\text{dim}}$, $F_J(x) = \sum |round(x_i)|$

Parabolic function If $x = (x_1, x_2, \dots, x_{\text{dim}}) \in [-500, 500]^{\text{dim}}$, $F_P(x) = \sum x_i^2$

Rosenbrock function For $(x, y) \in [-2, 2]^2$ $F_R(x, y) = 100(x^2 - y)^2 + (1 - x)^2$ whose minimum 0 is reached in a "banana valley shape" at (1, 1).

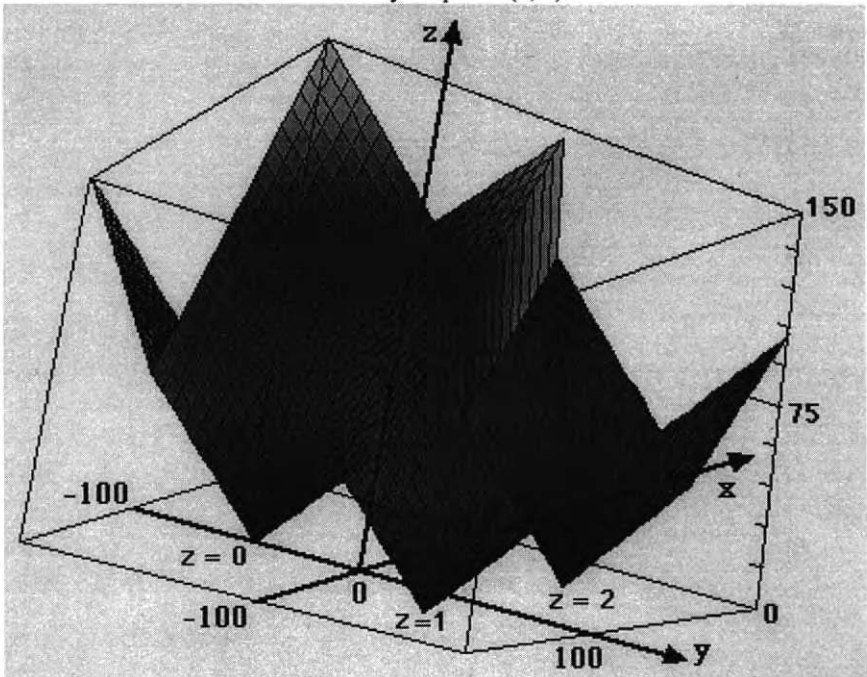


Figure 1. Artificial 2-dimensional "tripod" function (3 attractive minima, $\text{dim} = 2$ for $x, y \in [-100, 100]$) :
 $f(x, y) = \text{if } y < 0 \text{ then } |x| + |y + 50| \text{ else if } x < 0 \text{ then } 1 + |x + 50| + |y - 50| \text{ else } 2 + |x - 50| + |y - 50|$

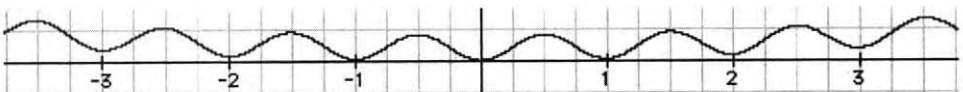


Figure 2. Rastrigin function $F_R(x) = 0.01[\sum_{1 \leq i \leq \text{dim}} x_i^2 + 10 - 10 \cos(2\pi x_i)]$ for $|x_i| < 500$, here $\text{dim} = 1$

Griewank function $x \in [-500, 500]^{\dim}$, $FG(x) = (1/4000)\sum_{1 \leq i \leq \dim} x_i^2 - \prod_{1 \leq i \leq \dim} \cos(x_i/\sqrt{i}) + 1$

Comparison criteria Usually the performance measure may be defined as a percentage at the time when the global optimum is reached when the algorithm has been executed 20 or more times (number of generations being bounded). This does not measure the convergence speed but only the ratio of success. The fitness may be not very basic, but quite long to compute, so the main criterion would be the number of fitness evaluations. As we already know, here, the best fitness 0, it is possible to count the evaluation number to reach some bound as 10^{-6} . Another way, more interesting for unsupervised problems, is to fix this number and compare the best fitness reached by different methods. We chose the first way, averaging the results on 20 random runs, with a restriction : a maximum of evaluations max is set to 10^5 .

4. Results

View of previous comparison trials between $ES(\mu+\lambda)$, $ES(\mu, \lambda)$ and elitist-ESAO

The following tests are always done with the same representation; we show the number of evaluation to reach the same constraint. Results for size 10 or 50 are in the same size order. Here the strategy $ES(\mu+\lambda)$ is used with (1 to 9) operators of our family, randomly applied to each father to produce from 1 to 9 offspring. Results show a confirmation for the optimal rate $\lambda/\mu = 7$, with a sensible amelioration with ESAO.

Rate λ/μ for ES	ESAO	ES 1	ES 2	ES 3	ES 4	ES 5	ES 6	ES 7	ES 8	ES 9
De Jong dim = 3	579	747	745	721	863	910	836	689	811	803
... .. dim = 5	897	2311	1832	3869	1823	2140	1844	1145	1276	1460
... .. dim = 10	1996	10072	11327	6450	4698	5517	5580	2599	2896	3005
Rosenbrock dim = 2 $\mu = 10$	12114	18471	15698	15746	16036	15645	14746	13985	15977	14059
... .. $ES(\mu, \lambda)$	18628	16407	19105	17864	16450	16210	15667	14857	13610	15917
Parabola dim = 3 $\mu = 10$ elitist	2114	16628	15795	14081	max	19050	max	6856	7610	6341
... .. $ES(\mu, \lambda)$	max	max	19634	12025	1932	2220	2305	3667	5942	5846
... .. dim = 5 $\mu = 10$	3083	19063	19395	18988	max	max	max	10282	11430	13595
... .. dim = 5 $\mu = 50$	6857	16935	16185	18822	19170	19337	max	11057	12210	14427
... .. dim = 10 $\mu = 50$	25302	max	max	48507	48190	49225	max	23990	21310	28670
Rastrigin dim = 10 $\mu = 50$	11141	26936	49874	47931	48883	49270	50207	16831	17531	14632
Griewank dim = 5 $\mu = 50$	49821	max	max	max	max	max	max	71927	max	max
... .. dim = 10	72027	max	max	max	max	max	max	80579	max	max

Table 1. Averaging on 20 runs for the "evaluation number" < max to reach "best fitness" < eps. The two best results by row are in grey. For non elitist strategies $ES(\mu, \lambda)$, results were scattered and not very pertinent.

We compare with the criterion above, six different strategies with the same size $\mu = 100$ and the same encoding. First is a classical GA with two probabilities. In second and third columns we write ES 7 the $ES(\mu + 7\mu)$ where each parent produces 7 children either by crossing-over or by mutation, or with the list of all operators defined above. The first conclusion is that we get better results when a list of operators is used instead of only two, (using various operators is using various methods to explore the space). The SSGA($\mu = 100$, $\tau = 33\%$) (without elimination) presented here, is also applied with a list of 14 operators and has similar results as the elitist-ESAO. Results are roughly better and better from the left to the right and we conclude that the hybrid algorithm ss-ESAO($\mu = 100$, $\tau = 33\%$) gives the best results.

Strategy	GA $p_c = 0.5$ $p_m = 0.1$	ES 7 mutat.- cross.	ES 7 operators list	elitist-ESAO	SSGA $\tau = 33\%$	ss-ESAO $\tau = 33\%$
De Jong dim 2	13447	3529	2714	1099	1177	920
... .. dim 10	max	58699	21371	7918	13530	6366
Parabola dim 10	max	max	30709	16481	9910	8621
Tripod dim 2	87637	16066	13534	12806	16950	10303
Rosenbrock dim 2	95124	77925	53224	11520	22681	21713
Rastrigin dim 2	23221	9376	3566	2529	1540	1456
... .. dim 10	max	max	23386	14608	8697	7703
Griewank dim 2	16131	27975	8245	6435	3559	1106
... .. dim 10	max	max	74486	73590	90125	58723

Table 2. Number ($< \text{max} = 100\,000$) of function evaluations (averaging on 20 runs, population of size $\mu=100$)

Comparison of the ways to apply operators with a Steady State Algorithm 33%

We, now, show some results on a 100-sized population with an updating rate $\tau = 33\%$ between 4 ways to apply operators, the first one is the initial version of SSGA where each parent produces a child by mutation or crossing-over with a randomly picked other parent. In the second column, we test it with the list of the 14 operators above, more precisely the i -th child is created applying the $(i \bmod k)$ -th operator of the list containing k operators. In the third way, this operator is randomly picked in the list and the last column is the ESAO way.

Operators	mutation - crossover	list of operators	random operators	scored operators
De Jong dim 10	34995 (28374)	10104 (1377)	11021 (1424)	8703 (1404)
Parabola dim 2	2102 (1375)	1971 (1256)	1891 (1227)	2387 (1755)
... .. dim 10	65964 (30809)	17901 (4622)	17440 (3805)	16842 (5090)
Tripod dim 2	33504 (37502)	23971 (30770)	22679 (26740)	19529 (28089)
Rosenbrock dim 2	82261 (41008)	78607 (36138)	78022 (35565)	77287 (34440)
Rastrigin dim 2	2208 (1601)	2116 (1408)	1910 (1214)	2131 (1575)
... .. dim 10	71738 (31289)	16300 (2983)	16198 (2661)	12953 (3381)
Griewank dim 2	4243 (6066)	2779 (2609)	2533 (2450)	3738 (4364)
... .. dim 10	95235 (68756)	52234 (35386)	34358 (28294)	42550 (38529)

Table 3. Number of evaluation of the fitness (averaging on 100 runs) and standard deviation to reach a given threshold. Comparison for a fixed updating and clearing rate 33%

We conclude on an improvement from the left to the right and we have two remarks. As a run of one our strategy for a function $[a, b]^{\text{dim}} \rightarrow [0, M]$ tries to find a success (a solution x for $f(x) < \epsilon$), we can have an estimation of the number of evaluations Φ of f to reach this success. If p is the probability of this event, Φ could have a Pascal' distribution under the condition of independance of the runs and $E(\Phi) = 1/p$, $V(\Phi) = (1-p)/p^2$. Of course evaluations are not independent thanks to most of the operators, but if we look at the results, the standard deviation is on the order of the mean of Φ .

We can make some remark on favorite operators used during evolution, if we note that each time an operator is duplicated at the head of the list OP in the course of evolutions, we have for all those numeric applications crossovers favored :

crossover-3 (30899), crossover-0 (19460), crossover-2 (16638), crossover-1 (15074), symmetry (11518), mutation-1 (8794), migration-2 (7846), addition (7708), migration-all (6743), migration-3 (6673), copy (5503), mutation-2 (5212), migration-1 (3459), transposition (963).

In other classes of problems (research of fuzzy rules for fuzzy systems); the most disturbing operators as pruning, suppression of a rule ... were among the best ones [Gacôgne 94, 97]. Nevertheless, here migrations have a good score.

Steady State Algorithm ss-ESAO(100, τ , 33%) having regard on the updating rate τ

Updating τ	var	10	20	30	40	50	60	70	80	90	100
De Jong dim 2	1636	1232	1127	1112	1174	1273	1369	2180	2426	2433	5605
De Jong dim 10	15270	16236	9918	10246	10217	10516	10934	10985	1337	16915	max
Parabola dim 10	35850	35652	19276	19494	20947	22288	22643	27790	32367	45033	max
Tripod dim 2	22126	35304	22751	26002	24323	35107	25178	25393	21031	16384	54914
Rosenbrock dim 2	34333	46064	38373	30714	45289	28392	27336	24427	22634	38312	74749
Rastrigin dim 2	4829	7062	5236	3199	3606	4549	4111	5106	6580	7258	13707
... .. dim 10	30403	24898	18830	19461	17133	18445	18116	24849	28069	39147	max
Griewank dim 2	8092	11421	13114	8147	5608	6343	8557	8807	10387	12220	17130
... .. dim 10	82568	88418	74572	72193	73508	59363	66357	67020	74916	73488	max

Table 4. Number of function evaluations, averaging on 100 random runs to reach 10^{-6} (except Griewank 10^{-4}).

In this benchmark, the population size is always $\mu = 100$, $\max = 100000$. Clearing rate is $p = 33\%$ and operators are randomly taken. Results show that curiously we have 2 and sometimes 3 minima for updating rate between 10% and 90%, this will be confirmed later. Variable updating does not really bring performed evolution (first column), and the 100% rate as in standard GA loose elitist feature and brings bad results.

Steady State Algorithm $\tau = 33\%$ having regard on the clearing rate

Clearing	var	10	20	30	40	50	60	70	80	90	100
De Jong dim 10	9543	10272	10586	10616	10569	10187	8975	9266	9128	9295	8595
Parabola dim 2	2726	2325	2565	1597	1939	1848	2089	1880	1864	2405	1535
... .. dim 10	18989	17878	18643	21425	20254	20675	20754	19229	17876	20978	15430
Tripod dim 2	20752	14735	10610	15807	22906	29566	44932	32715	56042	49152	37480
Rosenbrock dim 2	80476	75387	86052	92764	83768	91730	71700	84841	90811	94646	76225
Rastrigin dim 2	2075	2172	1903	1800	1250	1605	1899	2111	2103	2400	1675
... .. dim 10	17044	18132	17605	19905	19466	16638	17494	16301	16124	18516	12040
Griewank dim 2	2682	1515	2410	3221	2175	3417	3168	2380	3214	3515	5625
... .. dim 10	82409	81151	76827	79194	63432	71435	79784	79717	76164	78183	84250

Table 5. Number of functions evaluations, averaging on 100 random runs to reach 10^{-6} (except Griewank 10^{-4}).

We intend to find an optimum rate for clearing, also for ss-ESAO(100, 33%, p) with a family of scored operators with rates p from 10% (very strong elimination) to 100% (any elimination). We could not have any conclusion about the clearing rate and we will see later, that it may be good roughly between $1/3$ or $2/3$. Sometimes 100% (no clearing) gives good results for convex functions but will be a dangerous way to stay inside a local minimum. Let us have a look at the swarm shape of the population, for the tripod function, repartition of P_t on $[-100, 100]^2$ during evolution :

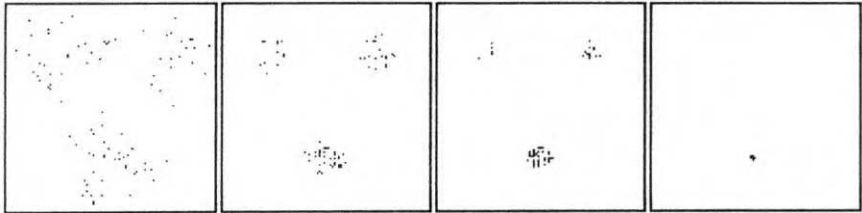


Figure 3. Population on $[-100, 100]^2$ $\mu = 100$ $\tau = 33\%$ without elimination for generations 1, 3, 5, 12. The population concentration is very quickly observed.

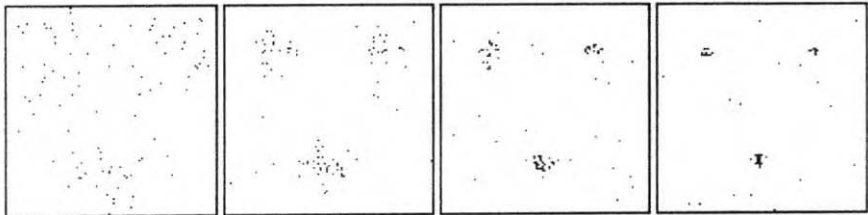


Figure 4. Generations 1, 3, 5, 12 with elimination rate $p = 33\%$. We remark for the last image a small group around the leader in the lower middle, but there is still an exploration favored by elimination.

Size μ of population for the ESAO(μ , 1/3, 1/3)-strategy

Pop. size μ	10	20	30	40	50	60	70	80	90	100
De Jong dim 2	523	758	1020	1152	1235	1428	2093	2184	1764	2030
... .. dim 10	2302	4310	4053	5308	6015	6642	7852	8304	8343	8860
Parabola dim 2	934	1240	1188	2060	1745	2412	2905	2392	3312	2810
... .. dim 10	3450	5758	6906	8292	8725	9480	9877	9792	10080	10000
Tripod dim 2	6978	6976	7035	7980	7930	9090	7658	8544	7056	8200
Rosenbrock dim 2	9316	8218	8111	9068	9205	8382	9450	9536	9099	8810
Rastrigin dim 2	1055	1556	1461	1720	2365	2286	2492	2864	2466	3540
... .. dim 10	3411	5406	6675	7784	9155	9378	9674	9920	10080	9950
Griewank dim 2	1190	2310	3747	6888	3895	2610	4830	4272	3870	4670
... .. dim 10	83510	82724	90828	87176	85290	92020	90014	89180	91080	87231

Table 6. Number of evaluation for $\tau = p = 33\%$ with different size of population

We can see here roughly increasing results according to the size of population, especially for the least difficult functions. Nevertheless, for the last one (10 dimensions Griewank function) and also for tripod and Rosenbrock functions, we get similar results for different sizes. This results show that very small populations are better, so we now want to have an idea about the whole triple (μ , τ , p) :

Evolutionary research for updating and clearing rates

In the above strategy called ESAO(μ , τ , p), let μ be the size of population, τ be the part of the best children population replacing the worst parents of each generation, and let p be the clearing rate (if a pair of individuals has similarity degree greater than p , the best is kept and the worst is removed by a new randomly chosen individual).
For any evolutionary algorithm, let us define $\Phi(\mu, \tau, p, f, a, b, \text{dim}, \varepsilon, \text{max})$ abbreviated $\Phi_{\mu, \tau, p}(f)$, the random variable equal to the number (bounded by $\text{max} = 100\,000$) of evaluations of a positive function f to reach $f < \varepsilon$ defined in the range $[a, b]^{\text{dim}}$. Let now (f_1, f_2, \dots, f_k) be a sequence of functions and :

$F(\mu, \tau, p) = (1/nk) \sum_{1 \leq i \leq k} \sum_{1 \leq j \leq n} \Phi_{\mu, \tau, p}(f_i)$ defined on $[1, 100]^3$ where $\Phi_{\mu, \tau, p}$ is the number of evaluations given by ss-ESAO(μ, τ, p). To optimize F , we use $\Phi_{500, 33, 66}$ as a meta-heuristic with 500 to have a sufficient visualisation, $1/3$ because one of the best empirical rate, $p = 2/3$ not too low to keep a compact swarm but still enough to get optimization. Because a quite large dispersion, $n = 100$ is preferred to $n = 20$ for averaging.

For test functions (f_1, f_2, f_3, f_4) we choose Tripod, Rosenbrock, Rastrigin and Griewank, each one with different parameters $[a, b]^{dim}$ for a benchmark based on easy or difficult functions to optimize the triple (μ, τ, p) .

So that no rigorous definition of the concept of difficulty is available, last trials allow us to choose the same functions in 10 dimensions, Griewank for $dim = 30, 100$ and tripod and Rosenbrock functions which are difficult to optimize.

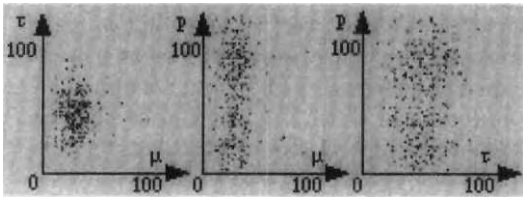


Figure 5. Three projections of an example of population of 500 triples (μ, τ, p) after 20 generations.

About twenty very long runs of $\Phi_{100, 1/3, 2/3}(F)$ with different combinations F and two runs of $\Phi_{500, 1/3, 2/3}(F)$ with Griewank and Rastrigin in 30-dimensions, provided always a population where μ is very small, between 3 and 12, a rate τ between 33% and 50% and p roughly around 33%. But for this last item we had often two swarms as above with also p around 66% for a worse part of population. So a last study of small populations is necessary and we did it with $p = 33\%$ which was the best pole.

Population size μ	$\mu=3$ $\mu\tau=1$	$\mu=4$ $\mu\tau=1$	$\mu=4$ $\mu\tau=2$	$\mu=5$ $\mu\tau=2$	$\mu=6$ $\mu\tau=2$	$\mu=7$ $\mu\tau=3$	$\mu=7$ $\mu\tau=4$	$\mu=8$ $\mu\tau=3$	$\mu=9$ $\mu\tau=3$	$\mu=10$ $\mu\tau=3$	$\mu=12$ $\mu\tau=4$
De Jong dim 2	556	537	551	526	415	463	618	700	558	617	759
... .. dim 10	2248	2167	2491	2136	1853	2046	2923	2600	2620	2616	2982
Parabola dim 2	1048	869	899	879	715	805	906	1077	733	805	972
... .. dim 10	2919	2724	3422	2755	2807	3439	4516	3848	3955	3831	4296
Tripod dim 2	6706	6971	6452	6315	4566	4274	5996	4650	7187	5451	11508
Rosenbrock dim 2	11766	11791	8705	9414	10011	8565	11016	8920	8820	9234	10479
Rastrigin dim 2	808	773	815	674	922	713	920	862	858	739	904
... .. dim 10	2886	2875	3423	3062	3064	2901	4185	3768	3553	3804	4414
Griewank dim 2	5360	5879	5081	5232	8678	2753	2522	2473	2692	4098	6256
... .. dim 10	40163	45879	12887	37466	25449	11836	13520	26105	40540	42775	53592
... .. dim 100	65417	62318	66686	59068	56765	55197	58292	59650	58609	60262	60780

Table 7. Number of evaluation (each one averaging on 100 runs) for $p = 33\%$ with small sizes of population.

Because of a large dispersion, we do not see meaningful differences, so, between population sizes $\mu = 3$ and $\mu = 12$, the only thing we could conclude is a recommendation to take $6 \leq \mu \leq 9$ with $\mu\tau = 3$ and $33\% < p < 66\%$ with a special mention for the case $\mu = 7, \mu\tau = 3$ in view of the results for the three most difficult functions among our trial.

5. Conclusion

As in other fields, models are first taken from nature to try to give a faithful but simple simulation, then nature is kept away in a second time. Different heuristics imagined around the paradigm of artificial evolution are more or less adapted to different problems. We assume that we have no means to choose the best representation, but when we have one, genetic operators can be defined in an intuitive way according to this representation.

The first conclusion of our study, for our representation and its operators, is an improvement of ES, even with the best empirical number of children 7, when a lot of various operators are randomly applied instead of only mutation and crossover. A second conclusion is, moreover, the benefit of a score attribution to those operators during the evolution (ESAO) to apply best operators to best individuals.

On the other hand, the good performance of SSGA in comparison with ES in the same conditions drove us to combine this heuristic with ESAO.

We think it is possible to explain those performances on two points: strategies where the best fitness is not decreasing along the generations, such as standard GA and similar, lose very often their best individual, strategies as $ES(\mu + \lambda)$ or elitist-ESAO where the average fitness is decreasing are too much elitist, they show a too homogeneous population. The aim of evolution algorithms in view of any optimization problem is not getting a good population but a good solution, and the best current solution is helped by the more or less good other solutions.

Strategies as SSGA where the averaging fitness is not decreasing, may give a sign of heterogeneousness. Two principles araised: to keep always the best or some different individual among the best and at the same time to explore by various methods during evolution with random migrations. So mixing SSGA and ESAO performs these principles.

The third conclusion is that a small size under 10 individuals provides better results, the best updating rate seems to be around 33% as also the best clearing rate.

References

- [1] Bäck T. *Evolutionary Algorithms in theory and practice*, Oxford University Press, 1995
- [2] Bäck T. Fogel D.B. Schwefel H.P. *Handbook of evolutionary computation*, Oxford University Press, 1997
- [3] Davis L. Adapting operator probabilities in genetic algorithm, *Proc. 5th Int. Conf. on GA* p61-69, Morgan K. 1993
- [4] De Jong K. Sarma J., *Generation Gaps Revisited*, in *Foundations of G.A.* Morgan Kaufmann, p5-17, 1993
- [5] Gacôgne L. Research of Pareto set by genetic algorithm, application to multicriteria optimization of fuzzy controller, *EUFIT* p.837-845, Aachen, 1997
- [6] Gacôgne L. Benefit of a steady state genetic algorithm with adaptive operators, *Mendel Conf Brno* p236-242, 2000
- [7] Goldberg D.E. *Genetic algorithms in search, optimization and machine learning*, Addison Wesley, 1989
- [8] Holland J.H. *Adaptation in natural and artificial system*. Ann Arbor University of Michigan Press, 1975
- [9] Kane C. Schoenauer M. Optimisation topologique de formes par AG, *Revue française de mécanique* 4 p237-246, 1997
- [10] Michalewicz Z. *Genetic algorithms + data structures= evolution programs*, Springer Verlag 1992
- [11] Schwefel H.P. *Systems analysis, systems design and evolutionary strategies*, System analysis, Modeling and Simulation vol.7 p.853-864, 1990
- [12] Schwefel H.P. *Evolution and optimum seeking*. Sixth generation computer technology series. Wiley, 1995
- [13] Tuson A. Ross P. Cost based operator rate adaptation, an investigation LCNS 1141 p461-469, PPSN 1996
- [14] Whitley D. Kauth J. Genitor : a different genetic algorithm, *Proc. of Rocky Mountain Conf. on A.I.* p118, 1988
- [15] Wolpert D. McReady W. No free lunch theorem for search, Research report, Santa Fe Institute (www.santafe.edu), 1995

Genetic Algorithms with Changing Criterion Functions

I. SEKAJ

*Department of Automatic Control Systems, Faculty of Electrical Engineering and
Information Technology, Slovak University of Technology, Ilkovičova 3*

812 19 Bratislava, Slovak Republic

Phone : +7 602 91 585

sekaj@kasr.elf.stuba.sk

Abstract. In practice there are many optimizing problems, where optimization criteria are change in time. In this paper an extension of the basic structure of genetic algorithm is proposed, which saves into an archive past solutions obtained after optimization under different environment conditions. This archive is used after each new change of the criterion. This “experience” from the past is able to considerably speed up the convergence of the genetic algorithm.

Keywords: genetic algorithm, time-changing criterion, archive of past solutions

1. Introduction

In many optimization tasks the requirements for optimal solutions are not constant all the time, but they vary in time. In other words, the criterion function (fitness function) parameters or their structure or structure of their constraints are changing in time. In such cases the used optimization method should be able to give the solution fast enough. In many applications the conditions are repeated more or less periodically or the new conditions are similar to those which occurred in the past. Therefore it is advantageous to save the old solutions – strings (chromosomes), which belong to past criterion functions, in an archive. If the conditions change, it is possible to use the old “experience”. A similar feature can be observed in the process of natural evolution, where the diploid chromosome structures and the role of dominance can preserve and transfer past information into the new generation. This mechanism has been employed also in the area of genetic algorithms [1], [2], [3], [4], [5], [6].

In this paper another new approach is proposed, which probably does not have its analogy in the nature. It is based on the archiving and utilization of all past solutions under different conditions (environments).

1.1 Archiving and utilization of past solutions

Consider the optimization task to be solved using the genetic algorithm (GA) formulated by the criterion function $J_k = f_k(x) \rightarrow \min$ and by the constraints $g_k(x) > 0$ and $h_k(x) = 0$, where $x \in R^n$ is the vector of potential solution, which is in the GA-structure coded into the string $x = [x_1, x_2, \dots, x_n]$ and k is the order number of the current change of conditions. By the condition change we mean an arbitrary change of parameters or structure of the criterion function f or of the constraints g or h .

this $(k+1)$ -st criterion. The best solution (or more solutions) with respect to the new conditions is then used in the third group of a strings (a can also equal 1) in each generation. The algorithm is repeated until the $(k+1)$ -st optimal solution is found.

Note: Another way of using the archive is to insert the best solution only once in the phase of the reinitialization of the population (dashed line in Fig.1).

2. Experiments

The proposed mechanism has been tested on some abstract mathematical minimization problems and on some practical applications. Let us demonstrate two examples. The first example is the minimization of the “Rastrigin function”. The second analyzed problem is a practical task of the optimal distribution of power generating between power plants in the electric power system.

2.1 Rastrigin function minimization

The Rastrigin function is defined as

$$f(x) = 10.n + \sum_{i=1}^n ((x_i - \xi_i)^2 - 10 \cos(2\pi(x_i - \xi_i)))$$

$$-5 < x_i < 5 ; \quad i = 1, \dots, n$$

It is a multimodal function of n variables, in our case $n=7$, where the searched global minimum is in the point $\xi = \{\xi_1, \dots, \xi_7\}$. Let the position of this point change periodically - it is „skipping“ between two areas, whereby inside these areas the points are located close to each other. The position of the global minima changes after each 300 generations as follows:

$$\begin{aligned} \xi_0 &= [0.1 \ 0.2 \ 0.3 \ 0.4 \ 0.5 \ 0.6 \ 0.7] \rightarrow \xi_{300} = [0.7 \ 0.6 \ 0.5 \ 0.4 \ 0.3 \ 0.2 \ 0.1] \rightarrow \\ \xi_{600} &= [0.15 \ 0.22 \ 0.34 \ 0.45 \ 0.51 \ 0.63 \ 0.71] \rightarrow \xi_{900} = [0.68 \ 0.59 \ 0.52 \ 0.42 \ 0.27 \ 0.22 \ 0.12] \rightarrow \\ \xi_{1200} &= [0.08 \ 0.19 \ 0.29 \ 0.39 \ 0.49 \ 0.58 \ 0.69] \rightarrow \xi_{1500} = [0.74 \ 0.64 \ 0.54 \ 0.44 \ 0.34 \ 0.24 \ 0.14] \rightarrow \\ \xi_{1800} &= [0.11 \ 0.22 \ 0.31 \ 0.41 \ 0.51 \ 0.62 \ 0.71] \rightarrow \xi_{2100} = [0.75 \ 0.58 \ 0.55 \ 0.35 \ 0.31 \ 0.25 \ 0.13] \rightarrow \\ \xi_{2400} &= [0.15 \ 0.24 \ 0.28 \ 0.43 \ 0.47 \ 0.62 \ 0.75] \end{aligned}$$

In Fig. 2 results are compared of the criterion function convergence for a standard GA without using the archive (marked “0”) and for the same type of GA using the archive of past solutions (marked “1” and bold). In the beginning both cases have converged approximately with the same speed. But later on, when there are solutions from neighbouring locations in the archive, the solution convergence will be much faster.

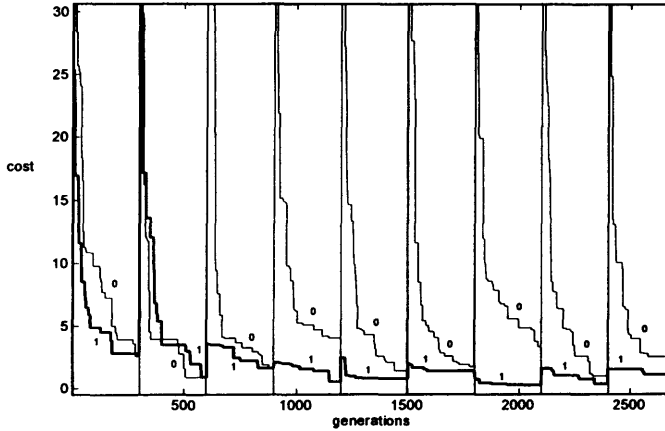


Figure 2. Convergence comparison of a standard GA (0) and the GA using the archive of past solutions (1, bold) for the Rastrigin function minimization

2.2 Optimal distribution of power

A practical area where the proposed method is preferably applicable is the problem of optimal distribution of effective-power between the sources in the electricity power system (source = turbine + generator). In the production of electric energy in each moment the sum of the generated power must be equal to the actual consumption. Therefore when the consumption changes in time (continuously or by jumps), the power production must also change. Each source in the electricity system has its own cost characteristic, which is defined as the dependency of costs for the production of 1 MWh on the power of this source in MW. The cost characteristics can be approximated by quadratic functions. The aim of the optimization is to constantly adapt the required power of all sources according to the current consumption in order to minimize the sum of all costs. The solution is the vector of all required values of power for each source, which represents the entries into the primary effective-power controllers in each particular source. The cost characteristics are non-continuous functions in the form

$$\begin{aligned} \text{if } P_{i,\min} < P_i < P_{i,\max} : C_i &= a_{i,2} P_i^2 + a_{i,1} P_i + a_{i,0} \\ \text{else } P_i &= 0, C_i = 0 \end{aligned}$$

where i is the number of the source, C_i are the costs of the i -th source [price/MWh] and P_i is the power of the i -th source [MW]. The solution represents such a combination of required powers of all sources for which the cost function is minimal

$$J = \sum_{i=1}^n C_i \rightarrow \min$$

and at the same time the condition

$$\sum_{i=1}^n P_i = L$$

is fulfilled, where L [MW] is the actual consumption in the entire power system. Possible disturbances are changes of consumption (ΔL) and also source-outage ($P_i \rightarrow 0$). In our

example $n=26$. Each 1000 generations a step-change of the consumption [MW] was realized as follows

3800→2200→3500→2500→3700→2300→3600→2400→3750→2250→
→3550→2420→3630→2370→3570

This problem has been solved through a conventional GA and compared with the proposed new one. In Fig. 3 convergence of the standard GA is marked "0" and the GA under use of the archive of past solutions is marked "1" with bold line. Again it is evident that after first steps of consumption (1000 and 2000 generations), when no past data are available, the convergence processes for both versions are comparable. But later on the access to past solutions starts to speed-up the convergence significantly.

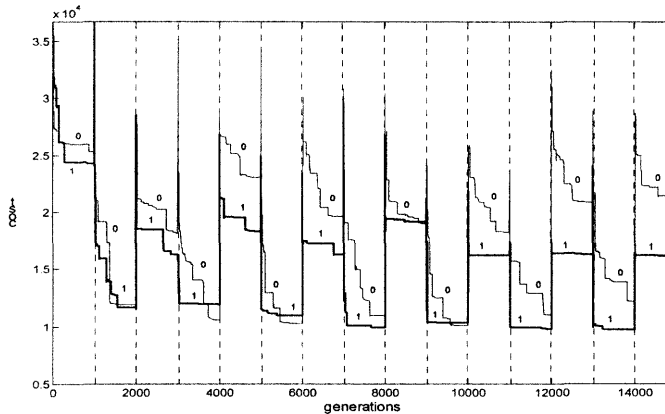


Figure 3. Convergence comparison of a standard GA (0) and the GA using the archive of past solutions (1, bold) for the power distribution problem

Let us remark that in this application, such a situation can occur due to frequent changes of consumption, that the change of the criterion function (change of consumption) can outrun the regular termination of the GA solution (after finding global optima). In that time the current (suboptimal) solution will be taken as the solution and a new search with new conditions will be started. This requires a modification of the GA structure from Fig.1, where the test of the "new criterion" will move into the inner cycle of the GA, behind the test of terminating conditions in the actual generation.

3. Conclusion

The paper presents a new method for genetic algorithms improvement when the conditions of the optimised environment change in time. The use of an archive of past solutions can considerably speed up the process of convergence of the GA, mainly when similar conditions (parameters of criterion function) occur. This approach can be useful also in cases, where only some partial optimization conditions are met, or only some partial features of the solution (some genes of the string) are identical or similar. There are many optimising problem types like this in practice and the realized experimental results have confirmed the advantages of the presented method.

Acknowledgement

This work has been supported within the VEGA grant No. 1/7630/20 "Intelligent methods of modelling and control" of the Slovak Grant Agency

References

- [1] D.E.Goldberg: Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley Publishing Company, Inc. 1989
- [2] V.Kvasnička, J.Pospíchal, P.Tiňo: Evolučné algoritmy, STU Bratislava 2000, in Slovak
- [3] Z.Michalewicz: Genetic Algorithms + Data Structures=Evolution Programs, Springer 1992
- [4] P.Ošmera, J.Roupec, R.Matoušek: Genetic Algorithms with Sexual Reproduction, proceeding of the conference SCI 2000, Orlando, USA, 2000, pp. 306-311
- [5] J.Roupec, P.Ošmera, R.Matoušek: The Behaviour of Genetic Algorithms in Changing Environment, proceeding of the conference Mendel 2001, pp. 84-90, 6.-8.6. 2001, Brno, Czech Republic
- [6] Z.Trojánek: Řízení elektrizačných soustav, FE ČVUT 1985, in czech

Visualization of Topic Distribution from Document Sequence on Web

Yasufumi Takama, Kaoru Hirota

Interdisciplinary Graduate School of Science and Engineering,

Tokyo Institute of Technology, 4259 Nagatsuta, Midori-ku, Yokohama 226-8502, Japan

takama,hirota@hrt.dis.titech.ac.jp

<http://www.hrt.dis.titech.ac.jp>

PREST, Japan Science and Technology Corporation (JST), Japan

Abstract. A method to find topic distribution from a sequence of document sets is proposed. Although the hugeness of the Web as well as its dynamic nature is a burden for the users, it will also bring them a chance for business and research if they can notice the trends or movement in the real world. The proposed clustering method employs the immune network model, in which the property of memory cell is used to find the topical relation among document sets. Several types of memory cells are proposed and compared with each other, and the effectiveness of the proposed method is also shown by applying to a sequence of online news articles.

Keywords: Topic distribution, Clustering, Web, Information visualization, Immune network

1 Introduction

A method to find topic distribution from a sequence of document sets is proposed. As the Web constantly provides us with new topics, it is difficult for the user to grasp the trends or change of topics on the Web. In particular, there are so many online-news sites on the Web and they constantly release up-to-date news articles of various topics. As another example, a series of user's retrieval processes also provides the user with a sequence of document sets. As noted in the timing problem[9], user's context can be lost easily through a series of information retrieval processes. Although the hugeness of the information resource as well as its dynamic nature is a burden for the users, it will also bring them a chance for business and research if they can notice the trends or movement in the real world, which cannot be found from a single document but from a sequence of document sets.

Information visualization systems[3, 12, 13, 14] are promising approaches to help the user notice the trends of topics on the Web. However, what the conventional systems such as Scatter/Gather[3] and Grouper[13, 14] are concerned with has been to process a single set of documents, and no sequential nature of document sets is considered. Although the clustering methods that gradually construct the structure [2, 5] have been already proposed, they do not intend to detect the topic stream.

In this paper, applying the *plastic clustering method*[8, 9, 10, 11] to find topic distribution from a sequence of document sets is proposed. The plastic clustering method is one of

the WWW information visualization systems that are based on document clustering method, and one of its characteristic features is the generation of *keyword map* as well as document clustering. Furthermore, the keyword's activation value is calculated based on the *immune network model* [1, 4, 6, 7], which is also useful as the visualization metaphor to improve the understandability of the keyword map.

In this paper, the model of *memory cell* is proposed and incorporated into the plastic clustering method, so that it can find the topical relation among different document sets. An Experimental result shows that the plastic clustering method with the proposed memory cell can find the topic stream from a sequence of online news-article sets.

The plastic clustering method is described in Section 2, and several types of memory cell models are proposed in Section 3, followed by the experimental results in Section 4.

2 Immune Network-based Plastic Clustering Method

2.1 Plastic Clustering Method

The plastering clustering method is proposed to generate a keyword map¹, while performing a document clustering. In the plastic clustering, all the documents in the same cluster have to contain at least one keyword, i.e., a *cluster identifier*. As the cluster identifier is expected to represent the significant topic found on the document set, it is also useful as a *landmark* on the keyword map that helps users to grasp the topic distribution over the document set. Therefore, the plastic clustering method can be viewed as the extraction method of a set of keywords (i.e., cluster identifiers = landmarks).

The algorithm of the plastic clustering method is as follows:

1. Extraction of keywords from a document set using the morphological analyzer² and the stopword list.
2. Construction of the keyword network by connecting the extracted keywords k_i to other keywords or documents d_i :
 - (a) Connection between keywords k_i and k_j :

Strong connection (SC): the number of documents D_{ij} containing both keywords is equal to or more than T_k .

Weak connection (WC): The D_{ij} is more than 0 and is less than T_k .
 - (b) Connection between k_i and a document d_j :

SC: the term frequency TF_{ij} of k_i in d_j is equal to or more than T_d .

WC: The TF_{ij} is more than 0 and less than T_d ³.
3. Calculation of keywords' activation values on the constructed network, based on the immune network model (Eq. (1)–(5)).

¹A keyword map is a 2-D space on which the keywords extracted from documents are arranged according to their similarities.

²As the current system is implemented to handle Japanese documents, Japanese morphological analyzer *Chasen*(<http://chasen.aist-nara.ac.jp/>) is used to extract nouns.

³ $T_k=3$ and $T_d=3$ are used in this paper.

4. Extraction of the keywords that activate higher than others as cluster identifiers.
5. Generation of document clusters according to the cluster identifiers.

As for the immune network model in Step 3, one of the simplest models proposed in the field of computational biology[1, 6, 7] is adapted.

$$\frac{dX_i}{dt} = s + X_i(f(h_i^b) - k_b), \quad (1)$$

$$h_i^b = \sum_j J_{ij}^b X_j + \sum_j J_{ij}^g A_j, \quad (2)$$

$$\frac{dA_i}{dt} = (r - k_g h_i^g) X_i, \quad (3)$$

$$h_i^g = \sum_j J_{ji}^g X_j, \quad (4)$$

$$f(h) = p \frac{h}{(h + \theta_1)} \frac{\theta_2}{(h + \theta_2)}, \quad (5)$$

here X_i and A_i are the concentration (activation) values of B-Cell i and antigen i , respectively. The s is a source term modeling a constant cell flux from the bone marrow and r is a reproduction rate of the antigen, while k_b and k_g are the decay terms of the antibody and antigen, respectively. The J_{ij}^b and J_{ij}^g indicate the connectivity between the antibodies i and j , and that between antibody i and antigen j , respectively. The influence on antibody i by other connected antibodies and antigens is calculated by the proliferation function (5), which has a log-bell form. The p is the maximum proliferation rate.

Applying a non-monotonic activation mechanism of immune network model enables us to satisfy the following contradictory conditions for a cluster identifier.

- A cluster identifier should connect to a certain number of keywords.
- There should not exist any connection between cluster identifiers.

Experiments are performed based on the questionnaires[11], and the results show that the quality of clusters generated by the plastic clustering method is comparable with or slightly better than the k-means clustering method.

Furthermore, as the cluster identifier suppresses the related keywords on the constructed keyword network, this relationship among keywords is also useful as the metaphor to improve the understandability of keyword map[11].

3 Introduction of Memory Cell for Context Preservation

3.1 Application of Plastic Clustering Method to Document-Set Sequence

The information visualization systems based on document clustering method assume that documents contained in the same set, i.e., the news articles released on the same day or the html documents retrieved with a single query, should have the relation to each other from a certain viewpoint. Furthermore, it is assumed in this paper that the document sets that are

Table 1: Parameter Settings for Memory Cells of Several Type

Type	k_b	p	θ_1	θ_2	R_{act}^i
Normal	0.4	1.06	10^3	10^6	$(607, 1.65 \times 10^6)$
TypeA	0.3	1.06	10^3	10^6	$(395, 2.53 \times 10^6)$
TypeB1	0.4	0.5	100	10^7	$(400, 2.50 \times 10^6)$
TypeB2	0.4	1.03	625	1.6×10^8	$(397, 2.52 \times 10^6)$
TypeB3	0.4	1.15	741	1.35×10^6	$(395, 2.53 \times 10^6)$

given sequentially have a certain relation to each other. Examples of a sequence of document sets are online news articles, which are released day by day, and that of the retrieval results obtained through a series of retrieval processes by a user.

In particular, we aim to find the following topic stream from a sequence of document sets.

- The topics that survive through several document sets correspond to the mainstream topics of the user's current retrieval task.
- When the topics, which were missed by the user in the early stage of browsing, appears again after several document sets, the user can reevaluate such missing topics based on the background knowledge obtained during the browsing processes.

To find mainstream topics and missing topics through a sequence of document sets, the same cluster identifier should be used for the similar topics contained in different document sets. By analogy with the immune system, such preferential keywords can be realized with the property of a *memory cell*.

To incorporate the property of a memory cell into the plastic clustering method, several types of memory cells are proposed in this section.

3.1.1 Memory Cell Model Based on Decay Term

In this model, the memory cell has a lower decay term than a normal antibody. The idea behind this model is that the activation region $R_{act}^i \subset R^+$ of an antibody i is defined as $R_{act} = \{h_i^b | f(h_i^b) > k_b\}$. Therefore, decreasing the decay term broadens the activation region. This type of memory cell is called *Type A* hereafter.

3.1.2 Memory Cell Model Based on Proliferation

The R_{act}^i can be broadened also by adjusting the value of θ_1 and θ_2 . When the interval between θ_1 and θ_2 increases, the peak of the proliferation function (Eq. (5)) also increases, which seems to affect the activation speed of the antibody. Therefore, several types of memory cells can be made in terms of the activation speed.

4 Experimental Results

4.1 Comparison of Memory Cell with Different Types

In this section, 4 types of memory cells (type A, B1, B2, and B3) introduced in Section 3 are compared with each other through experiments. The parameters used for memory cells, as

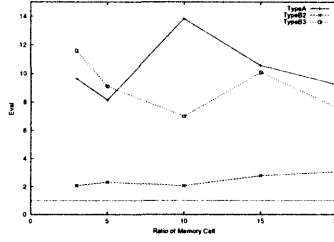


Figure 1: Comparison of Memory Cell and Normal Cell Keywords in Terms of Activation Tendency

shown in Table 1, are chosen so that their activation regions can be close to each other.

The purpose of the experiments is to show which type of memory cell is more robust than others in terms of activation, using the following measure.

$$Eval = \frac{|S_A \cup S_M|}{|S_M|} / \frac{|S_A - S_M|}{|S - S_M|} \quad (6)$$

where, S_A indicates the set of highly-activated keywords (i.e., cluster identifiers), S_M indicates the set of memory-cell keywords (hereafter, simply called 'memory cell'), and S indicates the set of all keywords.

The leading articles that were released from Yahoo! Japan News⁴ on Aug 8, 2001 are collected for the experiments. Yahoo! Japan News consists of 8 categories: domestic, international, economics, sports, entertainment, computers, industries, and business. Each category contains 5 leading articles in a day, and totally 40 articles are collected. From 40 articles, the keywords appearing in 3 or more documents are extracted. The number of extracted keywords is 115. In general, whether a keyword is highly activated or not can be determined based on its activation value, because the highly activated keywords have much higher (about 100 times higher) activation values than others[8].

The experiments are performed with several ratios of memory cells: 3%, 5%, 10%, 15%, and 20%. The numbers of memory cells are 3, 5, 11, 17, and 22, respectively. The selection of memory cells is performed randomly.

Fig. 1 shows the comparison results, in which the average values of five trials are shown. It is noticed that the memory cell of type B1 is not shown in the figure, because no memory cell of this type can be highly activated in any experiments. The reason why no memory cell of type B1 can be highly activated is that its activation speed, which is determined by $(p - d)$, is slower, though it has a broader activation region than normal keywords.

It is confirmed from Fig. 1 that the memory cells of type A, B2 and B3 are given the priority to activation compared with normal keywords. However, the memory cell of type B3 can have less priority compared with type A and B2, because its activation speed is slightly slower than those of type A and B2.

4.2 Topic Streams Extracted from a Sequence of News Articles

The plastic clustering method equipped with the proposed memory (type A) cells is applied to a sequence of news article sets. The news articles contained in the same set are issued on

⁴<http://news.yahoo.co.jp/headlines/>

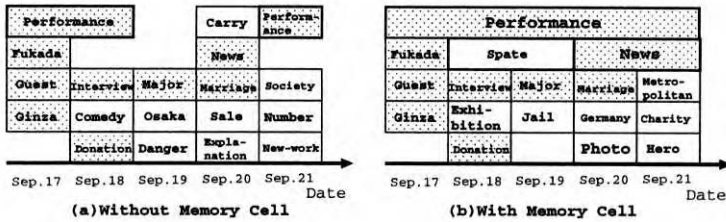


Figure 2: Cluster Identifiers Extracted from a News-Article Sequence (Translated from Japanese). The keywords extracted in both experiments are indicated with dotted texture, and topic stream is indicated with a thick border

the same day. News articles of entertainment topics that were issued from 17 September 2001 to 21 September 2001 are corrected from Yahoo! Japan News.

In Fig. 2, which shows the cluster identifiers extracted from a sequence of news articles, three topic streams, 'Performance', 'Spat', and 'News', can be found using memory cell, whereas only one topic stream can be found without memory cell.

The topic concerning the tragedy in N.Y. was a mainstream topic of the period when the target news articles were released. The plastic clustering with a memory cell can find the related topics ('Performance', 'Spat', etc.) from all document sets, while the clustering without a memory cell cannot find such topics from the document set issued on 19 September 2001.

5 Conclusion

A method to find topic distribution from a sequence of document sets is proposed. Four types of memory cell are proposed to find the relation among the clusters of different document sets, and experimental results show 3 types of those memory cells can have activation tendency compared with a normal cell. The plastic clustering method with memory cell is applied to a sequence of online news articles, and the results show a memory cell can play a role to find the topic stream through a sequence of document sets. The proposed method is expected to be a basis for the information visualization systems that handle a sequence of document sets.

References

- [1] R. W. Anderson, A. U. Neumann, A. S. Perelson, "A Cayley Tree Immune Network Model with Antibody Dynamics," *Bulletin of Mathematical Biology*, 55 (6), pp. 1091-1131, 1993.
- [2] D. H. Fisher, "Knowledge Acquisition Via Incremental Conceptual Clustering," in *Machine Learning*, 2, Kluwer Academic Publishers, pp. 139-172, 1987.
- [3] M. A. Hearst and J. O. Pedersen, "Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results," *Proc. 19th Int'l ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'96)*, pp. 76-84, 1996.
- [4] N. K. Jerne, "The Immune System," *Sci. Am.*, 229, pp. 52-60, 1973.
- [5] K. Lagus, T. Honkela, S. Kaski, T. Kohonen, "Self-Organizing Maps of Document Collection: A New Approach to Interactive Exploration," *2nd Int'l Conf. on Knowledge Discovery and Data Mining*, pp. 238-243, 1996.
- [6] A. U. Neumann and G. Weisbuch, "Dynamics and Topology of Idiotypic Networks," *Bulletin of Mathematical Biology*, 54 (5), pp. 699-726, 1992.

- [7] D. J. Smith, S. Forrest, A. S. Perelson, "Immunological Memory is Associative," Int'l Workshop on the Immunity-Based Systems (IBMS'96), 1996
- [8] Y. Takama and K. Hirota, "Application of Immune Network Model to Keyword Set Extraction with Variety," 6th Int'l Conf on Soft Computing (IIZUKA2000), pp. 825-830, 2000.
- [9] Y. Takama and K. Hirota, "Consideration of Presentation Timing Problem for Chance Discovery," 5th World Multiconference on Systems, Cybernetics and Informatics (SCI2001), 8, pp. 429-432, 2001.
- [10] Y. Takama and K. Hirota, "Employing Immune Network Model for Clustering with Plastic Structure," 2001 IEEE Int'l Symp. on Computational Intelligence in Robotics and Automation (CIRA2001), pp. 178-183, 2001.
- [11] Y. Takama and K. Hirota, "Immune Network-based Clustering for WWW Information Gathering/Visualization," Proc. of Japanese Society of Artificial Intelligence, SIG-FAI/KBS-J, pp. 61-66, 2001 (written in Japanese)
- [12] Y. Takama and M. Ishizuka, "FISH VIEW System: A Document Ordering Support System Employing Concept-structure-based Viewpoint Extraction," J. of Information Processing Society of Japan (IPSJ), 42 (7), 2000 (written in Japanese).
- [13] O. Zamir and O. Etzioni, "Grouper: A Dynamic Clustering Interface to Web Search Results," Proc. 8th Int'l WWW Conference, 1999.
- [14] O. Zamir and O. Etzioni, "Web Document Clustering: A Feasibility Demonstration," Proc. SIGIR '98, pp. 46-54, 1998.

A Tool for Data Mining Support

Marek HUBAL, Peter BEDNÁR

*Dept. of Cybernetics and AI, Technical University of Košice,
Letná 9, 042 00 Košice, Slovakia
bednar@neuron.fei.tuke.sk*

Abstract. This paper presents basic ideas and some of the results of the GOAL1 project. Within this project a KDD (Knowledge Discovery in Databases) package has been designed and implemented (Paralic and Andrassyova, 1999). Based on the GOAL pilot applications (Kouba et al., 2001) two data mining (DM) tasks are supported, namely classification and prediction. This paper describes how prediction DM task is supported within the KDD Package. From algorithms available there, case-based reasoning (CBR) prediction DM module uses evolutionary algorithm for optimizing weights of neighbor cases used for prediction. This particular module is presented here together with experiments and results achieved.

Keywords: Knowledge discovery in databases, data mining, prediction, evolutionary algorithm, case-based reasoning

1. Introduction

The main objective of the GOAL project was to develop a generic framework both recognized by the research community and applicable in real world applications, which solves the general issues of GIS and DWH (Data Warehouse) interoperability, including DWH feeding, knowledge extraction, interpretation, and security concepts. The feasibility of this framework has been tested on 2 very different real world applications from the GIS domain using environmental sensor data of a water supply company and cultural data on historical monument visitors (Rauber and Paralic, 2000), allowing a real world evaluation of the framework.

Within this project a KDD (Knowledge Discovery in Databases) package has been designed and implemented at the Department of Cybernetics and Artificial Intelligence, Technical University of Kosice.

In the following section 2, an overall architecture of the KDD Package is presented. Section 3 describes CBR prediction DM module with optimization based on the evolutionary algorithm in greater details. Finally, experiments and results achieved for two different data sets (one benchmark set and the other one from real application data) are summarized in section 4.

2. Architecture

KDD Package has a modular structure (see Figure 1), where common parts of the system can be used by each of the specialized DM modules.

2.1 Data Access Module

Data access module serves for accessing database sources of various types (which can be a text file, DBase, Paradox or MS Excel table, any local or remote SQL database using a visual SQL wizard or data warehouse through MS Excel interface). Data is connected (basic statistical measures are provided at the same time without actual loading of the data. If the user decides to process connected data, (it is physically loaded into the main memory) and forms a so called *view*, which can be previewed in three levels of granularity – a list of *operations* performed on the actual view, a list of attributes with statistics (so called *quick view*) or a two-dimensional table of *data*.

2.2 Module for data pre-processing

Module for data pre-processing enables a user to visualize, browse, modify, transform, sample etc. connected data. Pre-processing techniques are divided into those which operate on rows and those which operate on columns of the view.

All possible operations on data are defined by means of plug-in modules. This makes it possible to add a new transformation operation on data (sampling, discretization, etc.) any time very easily.

2.3 Data Mining Modules

For each KDD task a different DM algorithm as well as knowledge visualization component is suitable. Therefore each new DM algorithm and its respective knowledge visualization component can be implemented separately and added into the KDD package in the form of a separate plug-in DM module. It does not necessarily mean that each DM algorithm must have its own knowledge visualization component.

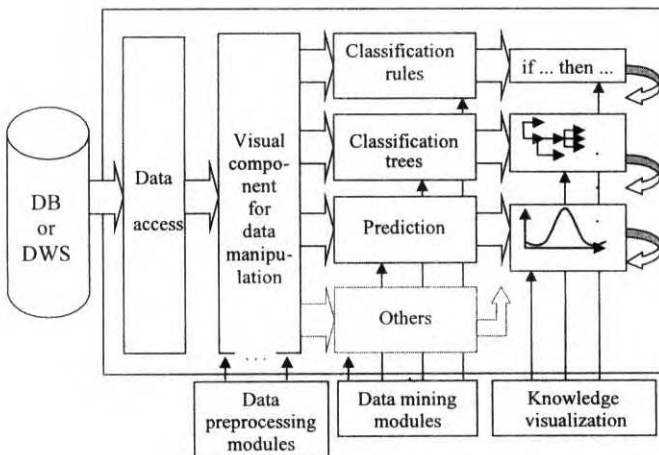


Figure 1. Architecture of the KDD Package.

Usually in order to add a new DM task functionality into the KDD Package the following steps need to be done.

- Implementation (or just re-use of an existing implementation) of a data-mining algorithm in the form of a plug-in module

- If necessary, implementation of new transformation or other pre-processing functions in the form of a plug-in module
- If necessary, implementation of a new knowledge visualization component in the form of a plug-in module.

Process of adding a new plug-in module may be controlled directly from the running KDD package application.

Based on character of the real data from the GOAL project pilot applications, classification and prediction DM tasks functionality have been implemented by means of various algorithms and their combination. KDD Package with its DM modules has been tested on the real data from two pilot applications as well as on various benchmark data sets.

3. Prediction

Prediction DM task has been used for prediction of water consumption based on environmental sensor data of a water supply company in western Bohemia. The following algorithms have been implemented:

- Linear regression
- MP5' producing regression trees (Witten and Frank, 2000)
- Case-based reasoning (CBR) approach where weights of the nearest cases taken into account for the calculation of the predicted value are optimized by means of the evolutionary algorithm.

3.1 CBR Method used

In the following our CBR approach will be briefly presented. A new problem is solved in CBR by finding a similar past case, and reusing it in the new problem situation. In case of prediction we use k nearest past cases to predict the target attribute of a new case. In order to find the k nearest cases, we use the following distance measure between two cases A and B :

$$D_{A,B} = \sum_{i=1}^n W_{A_i} |atr_i^A - atr_i^B| \quad (1)$$

where A is a past case, B a new case is the weight of attribute A , atr_i^A is the value of i -th attribute of the past case A , atr_i^B is the value of i -th attribute of the new case B and n is the number of attributes (without the target attribute to be predicted).

The attributes are normalized before calculating distances. If any of the attribute values is unknown, the value of the distance is set to 1. For nominal attributes if both attribute values are similar, their distance is 0, otherwise the distance is set to 1.

The value of the target attribute of a new case B is calculated as a weighted sum of corrected values of the target attribute of all k nearest past cases.

$$C = \sum_{i=1}^k W_{N_i} \cdot Corr_i C_i \quad (2)$$

where W_{N_i} is the weight of the i -th closest neighbor case, C_i is its value of the target attribute, $Corr_i$ is correction used for the i -th closest neighbor case and k is the number of neighbor cases taken into account for prediction. Correction used for the i -th closest neighbor case is calculated as follows.

$$Corr_i = 1 + \sum_{j=1}^n \left(1 - \frac{atr_j^A}{atr_j^B}\right) W_{C_j} \quad (3)$$

where W_{C_i} is the weight of the i -th attribute after correction.

3.2 *Weights optimization using evolutionary algorithm*

For the set of all training cases, minimal, maximal and average errors are calculated. The quality of prediction is given by the used weights (W_{Ai} , W_{Ni} , W_{Ci}) and the number of past neighbor cases (k) taken into account for prediction. A user may set up all these parameters. Weights are further optimized in order to achieve a minimal prediction error by means of the evolutionary algorithm.

Each individual in the population is represented as a one-dimensional vector of the real values of all weights W_{Ai} , W_{Ni} , W_{Ci} for $i = 1, \dots, n$. It is possible to choose between mean deviation and mean squared error for fitness function that defines the quality of each individual.

The user gives population size and may also specify initial weights, which will be used for the first (best) individual in the initial generation. Termination condition can be specified by means of a maximal error (minimal fitness) of the best individual by means of the number of generations or both.

Each new generation is calculated by using two genetic operators – mutation and crossover. There are two types of mutation, one that should produce individuals significantly different from the parents (random change of weights from predefined intervals) and the other one mainly aimed for fine changes in individuals (random change of weights in a very small interval). Crossover means that less than half of the weights are changed between two parent individuals. Particular genetic operators are applied to each parent individual with specific probabilities, which may be set up by the user.

New generation is selected from all individuals produced within a generation (all parents as well as all individuals newly generated by using genetic operators) as follows. All individuals are sorted by their fitness and 80% of the best plus 20% of the worst is taken into the new generation of individuals.

4. Experiments and results achieved

4.1 *Computer Performance Prediction*

As referenced benchmark dataset data about computer performance has been used (Ein-dor and Feldmesser, 1987). In this case, the predicted attribute is relative computer performance. Using our approach described in the previous section we achieved an average error rate 29.9%, which is significantly better than average error rate 34.1% achieved by the authors of this prediction problem in (Ein-dor and Feldmesser, 1987).

4.2 *Prediction of Water Consumption*

The real data application has been concerned with the prediction of drinking water consumption. We had available data from a water supply company in western Bohemia from the time period January 1999 - November 1999 and January 2000 - June 2000.

At first, using SQL scripts we transformed data from data warehouse into a single table. Data from one reservoir has been used for prediction. In the next step, we summed up all measurements achieved in one day in order to obtain values of "one-day" attributes.

Finally, we computed average values of attributes for some past days.

Data from year 1999 has been divided into two sets and data from year 2000 has been used as a testing set. Altogether we used 212 cases (examples) with 26 attributes. Predicted attribute was CONSUMPTION.

We have chosen 4 attributes with the highest correlation coefficients from different groups of attributes MONTH (time dimension, describes month of year, possible values 1 - 11), OUTFLOW_2 (from outflow attributes, describes water amount in m³ flowed out from the tank during previous two days), CONS_2 (cumulative consumption, water amount in m³ consumed from the tank by customers during previous two days), INFLOW_2 (from inflow attributes, water amount in m³ added to tank during previous two days). The lowest prediction error that we obtained on this set of attributes was 13.17%.

Next, we changed the used attributes as follows. OUTFLOW_2 and INFLOW_2 stayed. We removed attribute MONTH and replaced attribute of consumption CONS_2 with the most correlated one from the weather attributes - TEMP_15 (Average temperature in °C within previous 15 days).

The lowest prediction error that we obtained on this set of attributes was 10.86%, which is less than in the previous case.

References

- [1] Kolonder, J.: Case-Based Reasoning. Morgan Kaufmann Publishers (1993)
- [2] Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P.: The KDD Process for Extracting Useful Knowledge from Volumes of Data. Comm. of the ACM, 11 (1996), 27-34
- [3] Han, J., Kamber, M.: Data Mining – Concepts and Techniques. Morgan Kaufmann Publishers, (2000)
- [4] Kouba, Z. - Mikšovský, P. - Matoušek, K. On Geographical On-Line Analytical Processing (GOLAP). In: World Multiconference on Systemics, Cybernetics and Informatics. Orlando (Florida) : International Institute of Informatics and Systemics, vol. 1 (2001), 201-205
- [5] Paralic, J., Andrassyová, E.: Intelligent Knowledge Discovery. In: Rudas, I.J., Madarasz, L. (eds.): Proc. from the 3rd IEEE International Conference on Intelligent Engineering Systems INES'99. Elfa Academic Press Ltd., Kosice (1999) 307-310
- [6] Rauber, A., Paralic, J.: Cluster Analysis as a First Step in the Knowledge Discovery Process. In Journal of Advanced Computational Intelligence, Fuji Technology Press Ltd., ISSN 1343-0130, Vol. 4 (2000), No. 4, pp. 258 - 262
- [7] Witten, I.H., Frank, E.: Data Mining. Morgan Kaufmann Publishers, San Francisco, California (2000)
- [8] Phillip Ein-dor and Jacob Feldmesser: Attributes of the performance of central processing units: A relative performance prediction model, 1987

Effects of Cuing on Perceived Location of Auditory Sources

Norbert KOPČO and Barbara SHINN-CUNNINGHAM

*Hearing Research Center, Department of Cognitive and Neural Systems
Boston University, 677 Beacon St., Boston, MA 02215
{kopco,shinn}@bu.edu*

Abstract. This study explores auditory processing mechanisms underlying the computation of sound source location. A previous study examined whether a cue sound that indicated which left-right spatial hemifield to attend could improve localization accuracy (Kopčo, Ler and Shinn-Cunningham, 2001) in the same way that it decreases response latency (Spence and Driver, 1994). Results show that, in an ordinary room, an informative preceding auditory cue does not improve localization accuracy; however, the presence of a preceding cue causes consistent localization bias of the target stimulus for cue-target delays as long as 300 ms. The current paper presents acoustic analysis that examines the extent to which localization bias can be explained by purely acoustic effects (including room reverberation) as opposed to neural processing effects (e.g., see Carlile, Hyams and Delaney, 2001).

Keywords: *auditory computation, auditory perception, sound localization, psychoacoustics, auditory attention, head-related transfer functions, reverberation*

1. Introduction

One of the goals of computational intelligence is to determine how biological systems solve difficult problems in order to apply these approaches to technical applications. The way in which biological systems compute sound source location from the acoustic signals reaching the ears can be instructive in this way, because biological systems do a remarkably good job of determining sound locations from what are often ambiguous, noisy acoustic spatial cues.

Figure 1 shows a diagram that outlines the neural pathway involved in the computation of the sound source position. As shown in Figure 1, the neural pathway for extracting source location is very hierarchical, involving many stages of processing even before the level of the cortex. Basic spatial cues are first extracted in the brainstem before being integrated in the midbrain. The resulting signals then reach the cortex, which governs high-level spatial behaviors. In addition to the feed-forward paths shown in Figure 1 (see also Shinn-Cunningham, 2001), there are complex temporal interactions and feedback in the spatial auditory pathway (which are just beginning to be characterized and are not included in Figure 1) that appear to aid the system in computing robust estimates of sound source location in the face of noise and uncertainty.

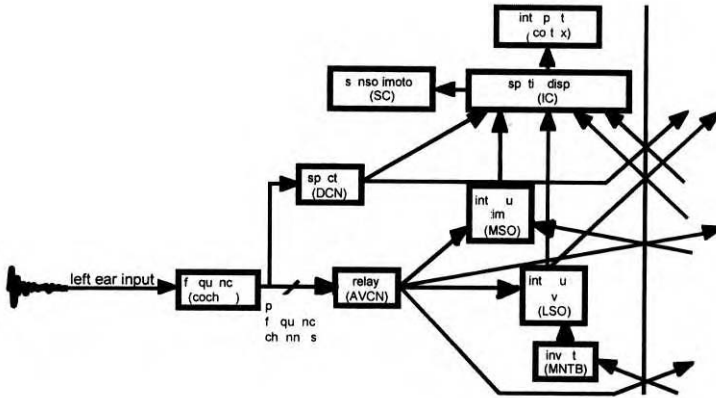
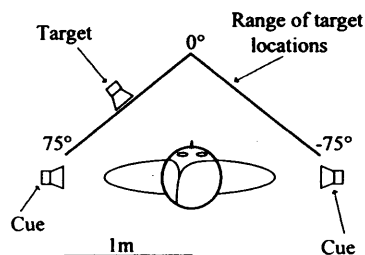


Figure 1. Simplified, block diagram of the forward neural pathway for computing source position (left half of symmetrical model; taken from Shinn-Cunningham, 2001). Abbreviations of physiological nuclei implicated in spatial auditory processing: DCN - dorsal cochlear nucleus; AVCN - anteroventral cochlear nucleus; MNTB - medial nucleus of the trapezoid body; LSO - lateral superior olive; MSO - medial superior olive; IC - inferior colliculus; SC - superior colliculus. Computations change over time (on many time scales) in order to maintain accurate spatial perception and include complex temporal dynamics, particularly at and above the level of the IC.

The goal of the present paper is to examine the extent to which simple acoustic interactions, rather than neural processing mechanisms (particularly temporal interactions), explain behavioral data obtained in a previous experiment studying cued auditory localization in a reverberant room (Kopčo et al., 2001).

Kopčo and colleagues investigated whether automatic and strategic auditory attention (Spence and Driver, 1994) influence human sound localization. The study tested whether the presence of an informative cue activates automatic and/or strategic auditory attention and improves sound localization accuracy (especially given that it reduces reaction times; Spence and Driver, 1994; and increases sensitivity in detection tasks; Sach, Hill and Bailey, 2000). Listeners indicated the azimuth angle of a target sound in the presence of a preceding cue stimulus that provided different amounts of information about the target location from block to block (see Figure 2). Results, which are summarized in Figure 3, show that the preceding cue generally degraded localization accuracy, independent of the amount of information provided by the cue, even when the stimulus onset asynchrony (SOA) between the cue and the target stimulus was 300 ms.

Figure 2. Experimental setup from Kopčo et al. (2001). In every trial, a cue sound from either +90° or -90° azimuth preceded a target sound positioned between +75° and -75°. Four experimental conditions were explored, varying the percentage of trials in which the cue correctly indicated the target hemifield (0, 50, 75, or 100%). The experiment was conducted with the listener located in the center of a quiet room (5 m x 9 m, T₆₀= 650ms).



There are two possible explanations for the observed localization bias due to the preceding cue. First, in a reverberant room, acoustical interactions between the cue and the target stimuli could cause systematic distortions of the acoustic spatial cues due to the target stimulus (such as interaural temporal differences or ITDs between the left and right ears). Such physical effects represent cue-target interactions that are external to the listener (i.e., that arise before the processing shown in Figure 1). Alternatively, the preceding cue may influence the internal, neural processing of the subsequent target (Figure 1). Such neural

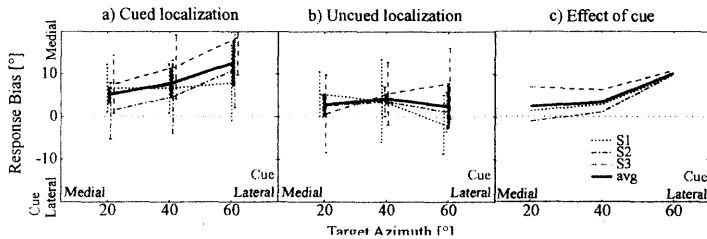


Figure 3. Effect of the cue on perceived target azimuth (from Kopčo et al., 2001). Each panel plots the mean and standard deviation in results for three individual subjects as well as an across-subject average. a) Localization bias (mean signed localization error) of the target in the presence of a cue. b) Localization bias of the target in isolation. c) Effect of cue on localization bias (difference in bias for cued and uncued conditions).

interactions could come about through temporal dynamics in the midbrain, or at higher stages of processing such as the cortex. The present study analyzes whether simple acoustic interactions can explain the effects of a preceding cue on localization of a target in order to rule out simple, parsimonious explanations for the observed behavior.

2. Methods

Acoustical interactions of the cue and target are investigated by examining the total signals reaching the ears of the listener for spatial configurations like those used in the behavioral study. This analysis focuses on how the energy due to the preceding cue source distorts the ITD in the onset of the target sound, which is assumed to be the dominant acoustical cue for horizontal auditory localization (Middlebrooks and Green, 1991). In order to simulate the total stimuli received at the ears of the listeners, the signals that would reach the listener from cue and target stimuli at appropriate locations are simulated using reverberant head-related transfer functions (HRTFs) measured in the room used in the study. Such HRTFs, which can be used to generate the total signal reaching the ears of the listener for an arbitrary source at a particular location in the room (e.g., see Shinn-Cunningham, Desloge and Kopčo, 2001), embody the acoustical interaction of the sound with the room and with the listener's body and pinna.

The HRTFs used in this study were measured on a Knowles Electronic Manikin for Acoustic Research (KEMAR) positioned in the same room as in the behavioral study. A 20-ms-noise-burst cue from $+90^\circ$ was simulated 50 ms before a 2-ms-click target stimulus from $+75^\circ$. This particular condition was chosen for analysis because the acoustic effects of the cue should be greatest for this condition compared to the others used in the behavioral study. In particular, for this condition, the cue and the target stimulus are in closer spatial and temporal proximity than in other conditions and the cue energy is relatively large compared to the target stimulus. If acoustic interactions are too small to explain localization bias for this configuration, they will be too small to explain bias in other configurations,

suggesting that the preceding cue influences how subsequent signals are processed in the spatial auditory pathway.

Left and right ear stimuli were bandpass filtered between 0.2 and 2 kHz and windowed through a 20-ms-wide Hanning window. The resulting 20-ms-long signal snippets were then cross-correlated to show how ITD varied during the course of the total cue-target stimulus. In order to get a better picture of how the ITD due to the target is affected by the

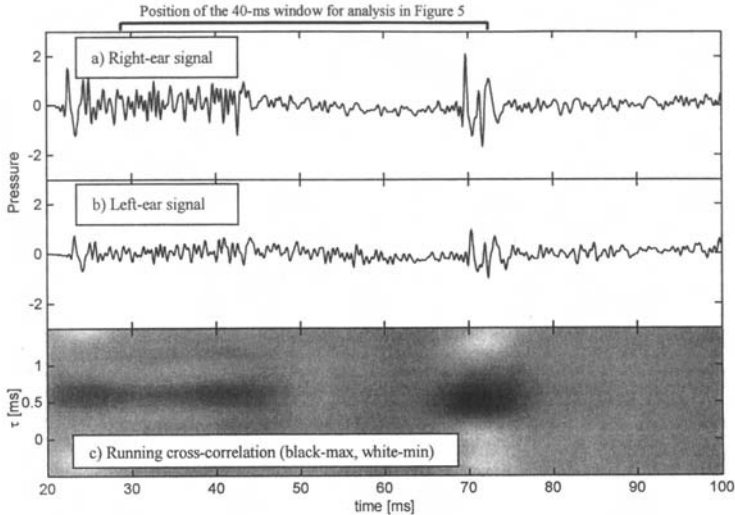


Figure 4. Signals received at the ears of the listener for a click-target at 75° and a noise-burst cue at 90° (50 ms later). a) The acoustic pressure signal received at the right ear. b) The acoustic pressure signal received at the left ear. c) The running, broadband cross-correlation of the left and right ear signals.

presence of the cue, ITDs were examined in detail at the onset of the target. In this analysis, interaural phase differences (IPDs) between the left and right ear signals were computed as a function of frequency in a 40-ms-long time slice containing the end of the cue stimulus and the onset of the target stimulus. This computation was also performed for the target signal in isolation. In both cases, the raw IPDs were calculated by dividing the left and right ear phase spectra; these IPDs were then converted into the corresponding ITD by dividing by frequency.

3. Results

Figure 4 shows both the left and right ear pressure waveforms at the ears (panels 4a and 4b, respectively) and the running cross-correlation of the left- and right-ear signals (panel 4c). In Figure 4c, the cross-correlation energy is plotted in grayscale; the black elongated region on the left corresponds to ITD in the cue and the second energy peak at approximately 70 ms corresponds primarily to the ITD in the target stimulus. These two cross-correlation peaks are clearly separated, despite the fact that the reverberant energy due to the cue is still relatively strong at the onset of the target stimulus (e.g., see panels 4a and 4b). The peak in the cross-correlation function corresponding to the target stimulus occurs at a slightly smaller ITD than that of the cue (lower value of τ along the vertical dimension), corresponding to the smaller ITD associated with target stimulus at 75° compared to the cue at 90°. In addition, the cross-correlation peak corresponding to the target stimulus is broader in the vertical dimension, due to the relatively short duration of the target stimulus compared to the width of the window used in the cross-correlation.

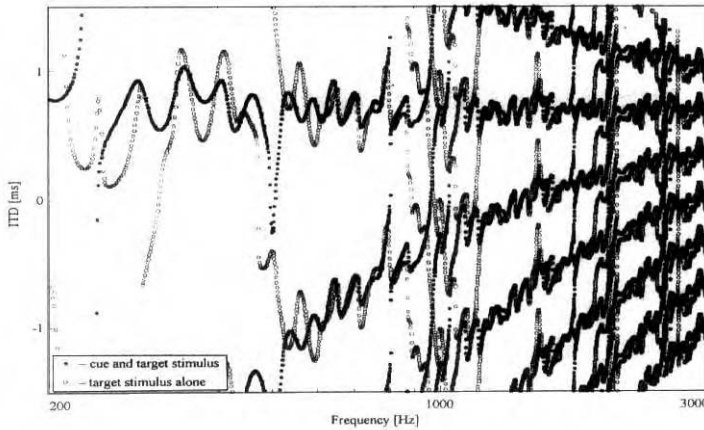


Figure 5. Effect of the cue on the ITD between left and right ear signals as a function of frequency in a 40-ms window centered on the onset of the target stimulus.

Figure 5 compares the ITD at the onset of the target in the cued and uncued conditions, where the interaction between the cue and the target stimulus should be greatest. The main effect of the cue shown in Figure 5 is an increase frequency-to-frequency variability in the ITD. However, this variation is also large for the uncued stimulus. More importantly, the ITD computed for the cued condition is not systematically biased towards 0, a result which could have explained the observed localization bias.

The results of the analysis shown in Figures 4 and 5 suggest that simple acoustic interactions of the cue and target stimuli cannot explain changes in the perceived target location due to the cue.

4. Conclusions and discussion

Given that 1) the information value of the cue has no effect on behavior, and 2) the localization bias due to the cue increases with the spatial proximity of the cue and target, attentional modulation cannot explain the observed effects of the cue on localization of the target. The current results also show that a simple bottom-up acoustic analysis cannot explain the behavioral data. This suggests that the observed effects (Kopčo et al., 2001) are due to a neural interaction between the representations of the cue stimulus and the subsequent target stimulus. Such effects may be related to short-term adaptation effects (e.g., see Kashino, 1998; Kashino and Nishida, 1998; Carlile et al., 2001). However, more central explanations may also play a role (e.g., see Sach, Hill and Bailey, 2000).

More thorough analysis of the acoustic interaction is needed to determine where in the auditory processing pathway interactions of cue and target arise. Specifically, analysis that considers both acoustical interactions and temporal interactions at low levels of processing (e.g., in the brainstem) may be capable of explaining these results. Such analysis is currently being undertaken to begin to pinpoint where in the human auditory pathway the response to a preceding cue stimulus influences the perception of a later-arriving target source.

References

- [1] Carlile S, Hyams S, Delaney S (2001) Systematic distortions of auditory space perception following prolonged exposure to broadband noise. *Journal of the Acoustical Society of America* 110:416-424.
- [2] Kashino M (1998) Adaptation in sound localization revealed by auditory after-effects. In: 11th International Symposium on Hearing.
- [3] Kashino M, Nishida Sy (1998) Adaptation in the processing of interaural time differences revealed by the auditory localization aftereffect. *Journal of the Acoustical Society of America* 103:3597-3604.
- [4] Kopčo N, Ler A, Shinn-Cunningham BG (2001) Effect of auditory cuing on azimuthal localization accuracy. *Journal of the Acoustical Society of America* 109:2377.
- [5] Middlebrooks JC, Green DM (1991) Sound localization by human listeners. *Annual Review of Psychology* 42:135-159.
- [6] Sach AJ, Hill NI, Bailey PJ (2000) Auditory spatial attention using interaural time differences. *Journal of Experimental Psychology: Human Perception and Performance* 26:717-729.
- [7] Shinn-Cunningham BG (2001) Models of plasticity in spatial auditory processing. *Audiology and Neuro-otology* 6:187-191.
- [8] Shinn-Cunningham BG, Desloge JG, Kopčo N (2001) Empirical and modeled acoustic transfer functions in a simple room: Effects of distance and direction. In: *IEEE Workshop on Applications of Signal Processing to Audio and Acoustics*. New Pfalz, New York.
- [9] Spence CJ, Driver J (1994) Covert spatial orienting in audition: Exogenous and endogenous mechanisms. *Journal of Experimental Psychology: Human Perception and Performance* 20:555-574.

Stability of Eco-Problem Solving in Positional Problems

Ivan KAPUSTÍK, Juraj HERCEK

*Department of Computer Science and Engineering, Slovak University of Technology,
Ilkovičova 3, 812 19 Bratislava*

kapustik@dcs.elf.stuba.sk, hercek@decef.elf.stuba.sk

Abstract. In this paper we present our study of eco-agent's behavior. We describe eco-problem solving our extensible simulation environment for positional and organizational problems. Eco-agent's behavior is illustrated on the experiment of cube problems. Our findings are analyzed and compared with similar works.

Keywords: *multi-agent systems, eco-problem solving, eco-agent, positional problems*

1. Introduction and problem definition

Eco-problem solving was invented in 1988 as an interaction of a set of simple agents. To define an eco-problem means to identify a population of agents (called eco-agents), whose set of behaviors tends to end in a stable state, which we call the solution of the problem (Ferber99). Each eco-agent tries to achieve its own goal individually. It transforms local perception into local actions. In addition to standard interaction actions, eco-agents can perform specific actions which, if taken sequentially, represent a plan how to reach a desired result.

The most important features specific to this approach are:

1. There is no global exploration of the space generated by the possible states of the world.
For this reason, this method involves very few combination explosions, which makes it capable of solving problems that are relatively large in size.
2. These systems resist disturbances very well. Disturbances are accepted by the system with no change in solving mechanism.

The first feature is well characterized in (Drogoul93) on the N-puzzle problem. Authors show that a locally optimized solution (eco-problem solving) is only about two times longer than a globally optimized solution (A algorithm). On the other hand, eco-agents were able to solve 899-puzzle in real time, while A algorithm only 15-puzzle.

The second feature emerges from reactivity of eco-agents. Disturbances are, of course, accepted only when they change a problem state, not the state or behavior of individual eco-agents (i.e. the disturbance does not change the behavior of the eco-agents, it changes only the relations between the eco-agents).

Though eco-agents have good features, as mentioned above, very little was done with them. First practical use of eco-problem solving was invented in (Sohier98). Their project motivated us to take advantage of eco-problem solving in our work. The aim of our project is to create an efficient simulation environment for organizational problems. This paper describes the first stage of the project (Hercek01), where we invented and created an extensible simulation environment and performed some tests.

2. Used methods and approaches

Eco-agents naturally divide actions into eco-interactive and problem specific ones. Architecture of our system tries to improve this advantage. It contains five main parts, depicted in:

1. Graphical user interface (GUI) – allows the user to choose a specific problem, appropriate output and to set initial problem settings.
2. ECO Core – implements eco-problem solving algorithm, this core can be fed by differently implemented plug-ins – problems themselves. ECO Core contains implementation of general behavior of eco-agents – functions `TrySatisfy()` and `TryEscape()`, see below.
3. Problem – implements specific problem (i.e. implementation of functions `Satisfy()`, `Escape()` and `FindLocationForEscape()`, see below.)
4. Output – provides diagnostic output information from ECO core and GUI
5. Problem Output – provides output information specific to the problem being solved

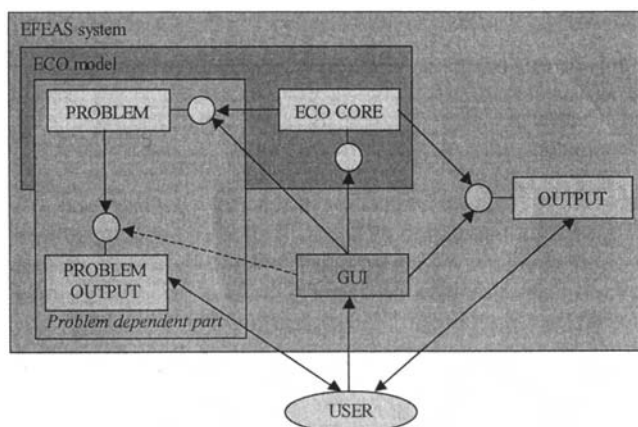


Figure 1. Main parts of EFEAS system

All parts of the system are designed as COM objects, which allow a problem to be implemented in any programming language supporting COM technology.

Now, we can look in more detail at the interactions of eco-agents. In general, actions can be coordinated arbitrary and can be implemented in different ways. Eco-agents represent the category of agents that are coordinated according to the specific rules. The coordination is defined by the means of eco-agent's behavior. Each eco-agent is characterized by the following behavior (Ferber99):

- The wish to be satisfied
- The obligation to escape

The *wish to be satisfied* forces the eco-agent to act in its environment towards the satisfaction of agent's own needs. Need is represented by the reaching of a certain state which is called a satisfaction state.

The environment typically consists of the set of eco-agents. A particular eco-agent can perceive other eco-agents as obstacles. In case a particular eco-agent perceives obstacle, it attacks it. The obstructor has the *obligation to escape*. The attacked agent perceives the attack of another agent and tries to escape – changes its state in a specific way in order to get rid of the attacking agent. The fleeing agent can attack another agent during the escape. This chain of attacking is broken down as soon as there is the eco-agent, which can achieve the state of satisfaction without attacking another eco-agent.

Eco-agent's *wish to be satisfied* can be described by the following pseudo code:

```

Procedure TrySatisfy(agent)
  If goal(agent) is satisfied and agent is not
  satisfied then
    For all obstructing agents do
      TryEscape(obstructor, agent, constraint)
    Satisfy(agent)
  End If
End Procedure

```

Procedure TrySatisfy(**agent**) is called for all agents sequentially or randomly – this depends on the implementation of the specific problem. The way of the selection of the eco-agents can be very important for some problems (Hercek01). Previous pseudo code specifies that the eco-agent's goal must be satisfied in order to satisfy a current eco-agent. This enables us to define dependencies – the sequence of satisfactions of all eco-agents. The goal of an eco-agent is another eco-agent whose state should be 'satisfied' in order another agent, its descendant, could try to satisfy itself.

Function Satisfy(**agent**) is depended on the specific problem and realizes the change of the state of the eco-agent, usually some movements of the eco-agents in their environment.

Procedure TryEscape(**obstructor**, **agent**, **constraint**) represents the second eco-agent's behavior – *obligation to escape*. Pseudocode of TryEscape function follows:

```

Procedure TryEscape(obstructor, agent, constraint)
  If obstructor was satisfied then obstructor
  becomes unsatisfied
    place = FindLocationForEscape(obstructor, agent,
  constraint)
    If place was not found then
      solution is not found - stop looking for solution
    and exit
    Else
      For all escape obstructing agents of obstructor
  agent do
        TryEscape(escapeObstructor, obstructor, place)
      Escape(obstructor, place)
    End Procedure

```

A constraint is very important in the function FindLocationForEscape(). This enables the algorithm to optimize output location for escape. FindLocationForEscape() is a problem dependent function. The place is usually a location in the environment and it can be used as a constraint in the context of the positional problems.

Procedure Escape(**obstructor**, **place**) is problem dependent and realizes the escape of the obstructor. Similarly like in Satisfy() procedure, this procedure causes the movement of eco-agents in the environment.

In order to understand eco-agents, we can describe them by the finite state automaton (Ferber99) – see Figure 2.

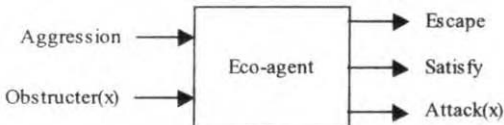


Figure 2. Eco-agent as a finite state automation

The eco-agent has two binary inputs represented by the aggression and the obstructor receptors. The output is represented by the taken action (Escape, Satisfy, Attack) and is determined by the input values received by the receptors and the internal state of the eco-agent. There are four internal states of the eco-agent:

- Satisfaction (S)
- Researching Satisfaction (RS)
- Researching Escape (RF)
- Escape (F)

All eco-agents are in the RS state at the beginning of the problem solving. Final state of the eco-agent is either S or RF if the solution is not found. Table 1 describes the transitions between the states.

Input			Output	
Current state	Aggression	Obstructor	Action	Next state
S	No	Do Not Care	No Action	S
S	Yes	Do Not Care	No Action	RF
RF	No	No	No Action	RS
RF	No	Yes	Attack	RS
RF	Yes	No	Escape	F
RF	Yes	Yes	Attack	RF
F	No	Do Not Care	No Action	RS
F	Yes	Do Not Care	No Action	RF
RS	No	No	Satisfy	S
RS	No	Yes	Attack	RS
RS	Yes	Do Not Care	No Action	RF

Table 1. Transitions between the states

3. Experiments and their results

Positional and organizational problems represent a huge group of problems that can be solved using eco-agents. This group of problems is characteristic by a starting and ending position description. We used a very simple example of positional problem during our study of eco-agents' behavior. It is a problem of sorting of cubes.

We have a table, which contains a limited number of free spaces. Cubes can occupy these spaces or they can be put one upon another – see Figure 3.

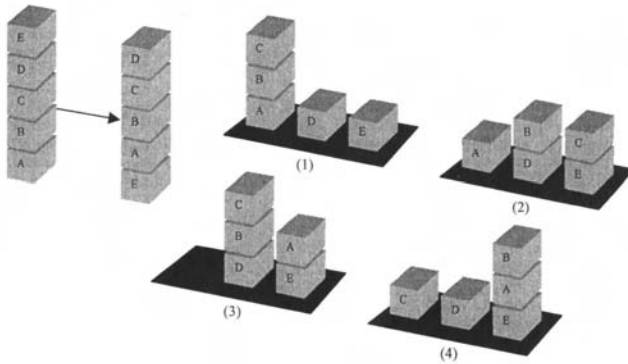


Figure 3. Problem of cubes

Each cube as well as free places on the table represent eco-agents. Left side of the picture describes the starting and ending position of the cubes and also dependencies between the cubes. The ending position implies dependencies – first satisfied cube (eco-agent) should be cube E, then A, B, C and finally D. The sorted set of dependencies = (E, A, B, C, D).

The right hand side of the picture describes the state of the eco-agents' environment during the search for the solution.

State (1) represents the state, when E satisfied itself and cube A attacked cube B, B attacked C and C attacked D which escaped on the free position on the table.

State (2) characterizes the state, when cubes B and C escaped up on cubes D and E, because they had finished their escape before cube A.

State (3) reflects the state when A attacked C and C escaped on the only possible position – cube B and A are satisfied as soon as it is free.

State (4) displays the state when cube B attacked C, which escaped on the free position on the table. Cube B satisfied itself. The final state is on the left-hand side of the picture.

We tried different starting and ending configurations and we received satisfying results – we found solutions, of course, they were not always the best ones.

4. Results analysis

Our experiments show two basic findings. The first finding is that solving of eco-problem is truly going on in the whole problem space. Even eco-agents, which are far from their goal, create clusters (in our case string of cubes) with their final-state neighbours. This 'subproblem solving' can either improve or worsen the efficiency of eco-problem solving. The second finding is that eco-problem solving needs well-defined dependencies of eco-agents.

5. Subproblem solving

We have found out that if we hadn't used constraint in *TrySatisfy* procedure, we would have received better results; the solution was found in smaller amount of moves. This improvement is sometimes quite significant – typically 12 moves instead of 16 in particular problem configuration – especially for problems with a lot of cubes and a few free places on the table.

This result is very interesting, because when we change the original problem into the problem of Hanoi towers by adding one ordering rule, *TrySatisfy* procedure must be written with constraint to achieve the best results.

Similarly, system has always found solution for and original problem when one existed. For the towers of Hanoi, the solution was granted only when *TrySatisfy* was used with constraint. Without constraint, agents could end in escaping cycles, when the first step was not optimal.

The explanation for the observed behavior is that strongly constrained eco-agents tend to create subsequences of the final solution on non-final places. Transformation from these subsequences to final state takes more time than transformation from more random sequences. Of course, these subsequences are necessary for the towers of Hanoi, while other approaches are inefficient.

This explanation is in very good agreement with N-puzzle experiment analysis in (Drogoul93). While eco-agents primarily choose their goal if it is adjacent, they progressively reduce the disorder of the puzzle during the solving, even when they are not trying to satisfy themselves. In fact, once the top row and the left column have been solved, the remaining puzzle is much less disordered than a random instance of the same size. All the movements generated by the first tiles have reduced the average distance of each unsatisfied agent to its goal. This emergent subproblem solving improves efficiency of N-puzzle solving. Unfortunately the same reason makes solving of problem of cubes much less efficient.

6. Dependencies of eco-agents

As we have already mentioned, satisfying of one eco-agent can depends on satisfying of other eco-agent. This dependency is usually derived from the final state of the problem. It also means that dependency is the only link which connects eco-agent's local view with global look at the problem and therefore it is the fundamental part of eco-problem solving. It determines the order of satisfying eco-agents and prevents endless escaping loops.

Many problems define dependencies not only from their final states. For instance, we can ask cubes to cross the bridge before they can reach their final position. Also, the order of cubes on the bridge is different than the one on the final position. Eco-agents, with their tendency to create partial solutions, will efficiently block the bridge if their dependencies are created only from the final state.

It is clear that previous example can be solved in two steps: first use dependencies for the bridge, then use dependencies for the final state. But some problems can produce different dependencies for different actual states or they are so complicated or vaguely defined that it is impossible to create right dependencies. Such problems can be solved by eco-agents too, but this is not recommended. Regular rebuilding of dependencies eliminates first eco-agents advantage – problem solving with only local information. Eco-agents must wait while global information is computed.

Incomplete or vague dependencies must be complemented by the random eco-agent selection. In this case the solution may not be found, even if more solutions exist. Also, when the solution is found it may not be quite optimal like for complete dependencies. Better results can be found for some problems when we add some artificial disturbances to system (Ferber99). In both cases, it may be necessary, to make a few problems run to find a solution.

7. Conclusion

We created an extensible simulation environment for positional and organizational problems, based on eco-problem solving. We used simple examples to study eco-agent's behavior, but these experiments revealed interesting relations between solution efficiency and stability and ordering in solved problems. Our first findings confirmed the results from (Drogoul93). Eco-agents are resistant to disturbances, they usually find solution close to optimal and they always tend to solve subproblems. Our other experience showed that subproblem solving could also weaken the efficiency of the complete problem solving. In addition we specified eco-agent dependency as a key feature of eco-problem solving and explained that the fundamental part of transformation from problem into eco-problem is the creation of the set of dependencies. This result allows a better decision about which problem is appropriate to solve with eco-agents.

The work reported here was partially supported by Slovak Science Grant Agency, grant No. G1/7611/20

References

- [1] DROGOUL A. and DUBREUIL C.: A Distributed Approach to N-Puzzle Solving. In: Proceedings of the 12th International Workshop on Distributed Artificial Intelligence. Pp. 95-108. 1993
- [2] FERBER J.: MultiAgent Systems. An Introduction to Distributed Artificial Intelligence. Addison-Wesley, 1999
- [3] HERCEK J.: Eco-agents. Thesis. Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Department of Computer Science and Engineering. 2001. www.hercek.sk/juraj/download/Thesis.zip
- [4] SOHIER C., DENIS B. and LESAGE, J.: Eco-problem solving for the adaptive control of production systems: the CASPER project. 9TH SYMPOSIUM of the International Federation of Automatic Control on INFORMATION CONTROL in Manufacturing systems, Nancy, Metz, France, June 24-26, 1998 (IFAC)

Particle Swarm Optimization Method for Constrained Optimization Problems

Konstantinos E. Parsopoulos

Michael N. Vrahatis

Department of Mathematics, University of Patras Artificial Intelligence

Research Center (UPAIRC), GR-26110 Patras, Greece

{kostasp, vrahatis}@math.upatras.gr

Abstract. The performance of the Particle Swarm Optimization method in coping with Constrained Optimization problems is investigated in this contribution. In the adopted approach a non-stationary multi-stage assignment penalty function is incorporated, and several experiments are performed on well-known and widely used benchmark problems. The obtained results are reported and compared with those obtained through different evolutionary algorithms, such as Evolution Strategies and Genetic Algorithms. Conclusions are derived and directions of future research are exposed.

Keywords: *Particle Swarm Optimization, Constrained Optimization, Swarm Intelligence*

1 Introduction

Constrained Optimization (CO) problems are encountered in numerous applications. Structural optimization, engineering design, VLSI design, economics, allocation and location problems are just a few of the scientific fields in which CO problems are frequently met [2], [4], [22]. The CO problem can be represented as the following nonlinear programming problem:

$$\min_x f(x), \quad x \in S \subset \mathbb{R}^n, \quad (1)$$

subject to the linear or nonlinear constraints

$$g_i(x) \leq 0, \quad i = 1, \dots, m. \quad (2)$$

The formulation of the constraints in Eq. (2) is not restrictive, since an inequality constraint of the form $g_i(x) \geq 0$, can also be represented as $-g_i(x) \leq 0$, and an equality constraint, $g_i(x) = 0$, can be represented by two inequality constraints $g_i(x) \leq 0$ and $-g_i(x) \leq 0$.

The CO problem can be addressed using either deterministic or stochastic methods. However, deterministic methods such as Feasible Direction and Generalized Gradient Descent, make strong assumptions on the continuity and differentiability of the objective function $f(x)$ [2], [4], [5]. Thus, there is an ongoing interest for stochastic algorithms that can tackle the CO problem effectively. Although Evolutionary Algorithms (EA) have been developed primarily as unconstrained optimization methods, they are considered as a good alternative for solving CO problems. Promising results have been reported during the past few years and

several variants of Genetic Algorithms (GA) [6], Evolutionary Programming [3], and Evolution Strategies (ES) [20], have been proposed to cope with the CO problem [7], [8], [12], [22].

The most common approach for solving CO problems is the use of a penalty function. The constrained problem is transformed to an unconstrained one, by penalizing the constraints and building a single objective function, which in turn is minimized using an unconstrained optimization algorithm [2], [4], [19]. This is most probably the reason behind the popularity of the Penalty Function approach when EAs are used to address the CO problem [8], [22].

In this paper, the performance of the Particle Swarm Optimization method (PSO) [1], [11], in solving CO problems is investigated. The CO problem is tackled through the minimization of a non-stationary multi-stage assignment penalty function. The results of experiments performed on well-known test problems are reported and discussed in comparison with results obtained by other EAs. In the next section, the Penalty Function approach is briefly described. In Section 3, the Particle Swarm Optimization method is presented, and, in Section 4, the test problems as well as the experimental results are reported. The paper ends with conclusions and ideas for future research, reported in Section 5.

2 The Penalty Function Approach

The search space in CO problems consists of two kinds of points: feasible and unfeasible. Feasible points satisfy all the constraints, while unfeasible points violate at least one of them. The Penalty Function technique, solves the CO problem through a sequence of unconstrained optimization problems [8]. Up to date, there exists no other method for defining pertinent penalty functions, than trial-and-error [22]. If the penalty values are high, the minimization algorithms usually get trapped in local minima. On the other hand, if penalty values are low, they can hardly detect feasible optimal solutions.

Penalty functions are distinguished into two main categories: stationary and non-stationary. Stationary penalty functions, use fixed penalty values throughout the minimization, while in contrast, in non-stationary penalty functions, the penalty values are dynamically modified. In the literature, results obtained using non-stationary penalty functions are almost always superior to those obtained through stationary functions.

A penalty function is, generally, defined as [22]

$$F(x) = f(x) + h(k) H(x), \quad x \in S \subset \mathbb{R}^n, \quad (3)$$

where $f(x)$ is the original objective function of the CO problem in Eq. (1); $h(k)$ is a dynamically modified penalty value, where k is the algorithm's current iteration number; and $H(x)$ is a penalty factor, defined as

$$H(x) = \sum_{i=1}^m \theta(q_i(x)) q_i(x)^{\gamma(q_i(x))}, \quad (4)$$

where $q_i(x) = \max\{0, g_i(x)\}$, $i = 1, \dots, m$. The function $q_i(x)$ is a relative violated function of the constraints; $\theta(q_i(x))$ is a multi-stage assignment function [7]; $\gamma(q_i(x))$ is the power of the penalty function; and $g_i(x)$ are the constraints described in Eq. (2).

The functions $h(\cdot)$, $\theta(\cdot)$ and $\gamma(\cdot)$, are problem dependent. In our experiments, a non-stationary multi-stage assignment penalty function was used. Details concerning the penalty function used in the experiments, are given in Section 4. In the next section, the PSO algorithm is described.

3 The Particle Swarm Optimization Method

PSO is a stochastic global optimization method which is based on simulation of social behavior. As in GA and ES, PSO exploits a population of potential solutions to probe the search space. In contrast to the aforementioned methods in PSO no operators inspired by natural evolution are applied to extract a new generation of candidate solutions. Instead of mutation PSO relies on the exchange of information between individuals, called *particles*, of the population, called *swarm*. In effect, each particle adjusts its trajectory towards its own previous best position, and towards the best previous position attained by any member of its neighborhood [9]. In the global variant of PSO, the whole swarm is considered as the neighborhood. Thus, global sharing of information takes place and particles profit from the discoveries and previous experience of all other companions during the search for promising regions of the landscape. To visualize the operation of the method consider the case of the single-objective minimization case; promising regions in this case possess lower function values compared to others, visited previously.

Several variants of PSO have been proposed up to date, following Eberhart and Kennedy who were the first to introduce this method [1], [10], [11]. The variants which were applied in our experiments are exposed in the following paragraphs.

Initially, let us define the notation adopted in this paper: assuming that the search space is D -dimensional, the i -th particle of the swarm is represented by a D -dimensional vector $X_i = (x_{i1}, x_{i2}, \dots, x_{iD})$ and the best particle of the swarm, i.e. the particle with the lowest function value, is denoted by index g . The best previous position (i.e. the position corresponding to the best function value) of the i -th particle is recorded and represented as $P_i = (p_{i1}, p_{i2}, \dots, p_{iD})$, and the position change (velocity) of the i -th particle is $V_i = (v_{i1}, v_{i2}, \dots, v_{iD})$.

The particles are manipulated according to the following equations (the superscripts denote the iteration):

$$V_i^{k+1} = \chi (wV_i^k + c_1r_{i1}^k(P_i^k - X_i^k) + c_2r_{i2}^k(P_g^k - X_i^k)), \quad (5)$$

$$X_i^{k+1} = X_i^k + V_i^{k+1}, \quad (6)$$

where $i = 1, 2, \dots, N$, and N is the size of the population; χ is a *constriction factor* which is used to control and constrict velocities; w is the *inertia weight*; c_1 and c_2 are two positive constants, called the *cognitive* and *social* parameter respectively; r_{i1} and r_{i2} are random numbers uniformly distributed within the range $[0, 1]$. Eq. (5) is used to determine the i -th particle's new velocity, at each iteration, while Eq. (6) provides the new position of the i -th particle, adding its new velocity, to its current position. The performance of each particle is measured according to a fitness function, which is problem-dependent. In optimization problems, the fitness function is usually identical with the objective function under consideration.

The role of the inertia weight w is considered important for the PSO's convergence behavior. The inertia weight is employed to control the impact of the previous history of velocities on the current velocity. Thus, the parameter w regulates the trade-off between the global (wide-ranging) and the local (nearby) exploration abilities of the swarm. A large inertia weight facilitates exploration (searching new areas), while a small one tends to facilitate exploitation, i.e. fine-tuning the current search area. A proper value for the inertia weight w provides balance between the global and local exploration ability of the swarm, and, thus results in better solutions. Experimental results imply that it is preferable to initially set the inertia to a large value, to promote global exploration of the search space, and gradually

decrease it to obtain refined solutions [21]. The initial population can be generated either randomly or by using a Sobol sequence generator [18], which ensures that the D -dimensional vectors will be uniformly distributed within the search space.

The PSO technique has proven to be very efficient for solving real valued global unconstrained optimization problems [13]–[17]. In the next section experimental results of the performance of PSO in CO problems are reported.

4 Experimental Results

The performance of three variants of PSO was investigated on well-known and widely used test problems. One variant utilizes only inertia weight (denoted as PSO-In), a second variant utilizes only constriction factor (denoted as PSO-Co), and the last one utilizes both inertia weight and constriction factor (denoted as PSO-Bo). For all variants, fixed values, considered as default, for the PSO's parameters were used: $c_1 = c_2 = 2$; w was gradually decreased from 1.2 towards 0.1; $\chi = 0.73$. Some variants of PSO, impose a maximum value on the velocity, V_{max} , to prevent the swarm from explosion. In our experiments V_{max} was always fixed, to the value of $V_{max} = 4$. The size of the swarm was set equal to 100, 10 runs were performed for each test problem, and the PSO algorithm ran for 1000 iterations, in each case. A violation tolerance was used for the constraints. Thus, a constraint $g_i(x)$ was assumed to be violated, only if $g_i(x) > 10^{-5}$.

Regarding the penalty parameters, the same values as the values reported in [22] were used, to obtain results comparable to the results obtained using different EA, in [22]. Specifically, if $q_i(x) < 1$, then $\gamma(q_i(x)) = 1$, otherwise $\gamma(q_i(x)) = 2$. Moreover, if $q_i(x) < 0.001$ then $\theta(q_i(x)) = 10$, else, if $q_i(x) \leq 0.1$ then $\theta(q_i(x)) = 20$, else, if $q_i(x) \leq 1$ then $\theta(q_i(x)) = 100$, otherwise $\theta(q_i(x)) = 300$. Regarding the function $h(\cdot)$, it was set to $h(k) = \sqrt{k}$ for Test Problem 1, and $h(k) = k\sqrt{k}$ for the rest problems.

The test problems are defined immediately below:

TEST PROBLEM 1, [4]:

$$f(x) = (x_1 - 2)^2 + (x_2 - 1)^2,$$

subject to $x_1 = 2x_2 - 1$, $x_1^2/4 + x_2^2 - 1 \leq 0$. The best known solution is $f^* = 1.3934651$.

TEST PROBLEM 2, [2]:

$$f(x) = (x_1 - 10)^3 + (x_2 - 20)^3,$$

subject to $100 - (x_1 - 5)^2 - (x_2 - 5)^2 \leq 0$, $(x_1 - 6)^2 + (x_2 - 5)^2 - 82.81 \leq 0$, $13 \leq x_1 \leq 100$, $0 \leq x_2 \leq 100$. The best known solution is $f^* = -6961.81381$.

TEST PROBLEM 3, [5]:

$$f(x) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7,$$

subject to $-127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leq 0$, $-282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leq 0$, $-196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leq 0$, $4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leq 0$, $-10 \leq x_i \leq 10$, $i = 1, \dots, 7$. The best known solution is $f^* = 680.630057$.

TEST PROBLEMS 4 AND 5, [5]:

$$f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141,$$

subject to $0 \leq 85.334407 + 0.0056858T_1 + T_2x_1x_4 - 0.0022053x_3x_5 \leq 92$, $90 \leq 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 \leq 110$, $20 \leq 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 \leq 25$, $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$, $i = 3, 4, 5$, where $T_1 = x_2x_5$ and $T_2 = 0.0006262$ for Test Problem 4, and $T_1 = x_2x_3$, $T_2 = 0.00026$ for Test Problem 5. The best known solution for Test Problem 4 is $f^* = -30665.538$, while for Test Problem 5 it is unknown.

TEST PROBLEM 6, [12]:

$$f(x, y) = -10.5x_1 - 7.5x_2 - 3.5x_3 - 2.5x_4 - 1.5x_5 - 10y - 0.5 \sum_{i=1}^5 x_i^2,$$

subject to $6x_1 + 3x_2 + 3x_3 + 2x_4 + x_5 - 6.5 \leq 0$, $10x_1 + 10x_3 + y \leq 20$, $0 \leq x_i \leq 1$, $i = 1, \dots, 5$, $0 \leq y$. The best known solution is $f^* = -213.0$.

For each test problem, the mean and the best solution obtained in all 10 runs, as well as the corresponding sum of violated constraints, were recorded. The results for all test problems are reported in Table 1. In all test problems, the three variants of PSO exhibited similar results.

Table 1: The mean and the best solution found in all 10 runs, for each method and problem. The standard deviation of the optimal values for the 10 runs is reported. In the parentheses the corresponding sums of violated constraints are reported

Problem	Method	Mean Solution (Sum V.C.)	St.D.	Best Solution (Sum V.C.)
1	PSO-In	1.394006 (0.000014)	0.0015	1.393431 (0.000020)
	PSO-Co	1.393431 (0.000020)	0.0	1.393431 (0.000020)
	PSO-Bo	1.393431 (0.000020)	0.0	1.393431 (0.000020)
2	PSO-In	-6960.866 (0.000037)	0.608	-6961.798 (0.000087)
	PSO-Co	-6961.836 (0.000019)	0.0011	-6961.837 (0.000019)
	PSO-Bo	-6961.774 (0.000013)	0.14	-6961.837 (0.000019)
3	PSO-In	680.671 (0.000008)	0.034	680.639 (0.000019)
	PSO-Co	680.663 (0.00034)	0.050	680.635 (0.00130)
	PSO-Bo	680.683 (0.000015)	0.041	680.636 (0.0)
4	PSO-In	-31526.304 (1.297)	18.037	-31543.484 (1.311)
	PSO-Co	-31528.289 (1.326)	12.147	-31542.578 (1.311)
	PSO-Bo	-31493.190 (1.331)	131.67	-31544.459 (1.311)
5	PSO-In	-31523.859 (0.958)	17.531	-31544.036 (0.997)
	PSO-Co	-31526.308 (0.965)	19.153	-31543.312 (0.996)
	PSO-Bo	-31525.492 (0.968)	23.392	-31545.054 (0.999)
6	PSO-In	-213.0 (0.0)	0.0	-213.0 (0.0)
	PSO-Co	-213.0 (0.0)	0.0	-213.0 (0.0)
	PSO-Bo	-213.0 (0.0)	0.0	-213.0 (0.0)

In most cases PSO outperformed the results reported in [22] for other EA. Proper fine-tuning of the PSO's parameters may result in better solutions.

5 Conclusions and Further Work

The capability of the PSO method to address CO problems was investigated through the performance of numerous experiments on well-known and widely used test problems. Preliminary results, obtained through the use of a non-stationary multi-stage penalty function,

imply that PSO is a good alternative for tackling CO problems. In most cases PSO detected superior solutions than those obtained through other EAs, as reported in [22]. It should be mentioned that the results were competitive in all cases, despite the fact that only the default parameters of PSO were considered. The performance of the three PSO's variants was similar for all test problems.

Future work will include investigation of the PSO's performance in other benchmark and real-life problems, as well as the development of specialized operators that will indirectly enforce feasibility of the particles and guide the swarm towards the optimum solution, as well as fine-tuning of the parameters that may result in better solutions.

References

- [1] Eberhart, R.C., Simpson, P.K., Dobbins, R.W.: Computational Intelligence PC Tools. Academic Press Professional, Boston (1996)
- [2] Floudas, C.A., Pardalos, P.M.: A Collection of Test Problems for Constrained Global Optimization Algorithms. Lecture Notes in Computer Science, Vol. 455. Springer-Verlag, Berlin Heidelberg New York (1987)
- [3] Fogel, D.B.: An Introduction to Simulated Evolutionary Optimization. IEEE Trans. Neural Networks 5(1) (1994) 3–14
- [4] Himmelblau, D.M.: Applied Nonlinear Programming. McGraw-Hill (1972)
- [5] Hock, W., Schittkowski, K.: Test Examples for Nonlinear Programming Codes. Lecture Notes in Economics and Mathematical Systems, Vol. 187. Springer-Verlag, Berlin Heidelberg New York (1981)
- [6] Holland, J.H.: Adaptation in Natural and Artificial Systems. MIT Press (1992)
- [7] Homaifar, A., Lai, A.H.-Y., Qi, X.: Constrained Optimization via Genetic Algorithms. Simulation 2(4) (1994) 242–254
- [8] Joines, J.A., Houck, C.R.: On the Use of Non-Stationary Penalty Functions to Solve Nonlinear Constrained Optimization Problems with GA's. Proc. IEEE Int. Conf. Evol. Comp. (1994) 579–585
- [9] Kennedy, J.: The Behavior of Particles. Evol. Progr. VII (1998) 581–587
- [10] Kennedy, J., Eberhart, R.C.: Particle Swarm Optimization. Proc. IEEE Int. Conf. Neural Networks. Piscataway, NJ (1995) 1942–1948
- [11] Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Morgan Kaufmann (2001)
- [12] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, New York (1992)
- [13] Parsopoulos, K.E., Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N.: Objective Function "Stretching" to Alleviate Convergence to Local Minima. Nonlinear Analysis TMA 47(5) (2001) 3419–3424
- [14] Parsopoulos, K.E., Plagianakos, V.P., Magoulas, G.D., Vrahatis, M.N.: Stretching Technique for Obtaining Global Minimizers Through Particle Swarm Optimization. Proc. Particle Swarm Optimization Workshop. Indianapolis (IN), USA (2001) 22–29
- [15] Parsopoulos, K.E., Vrahatis, M.N.: Modification of the Particle Swarm Optimizer for Locating All the Global Minima. V. Kurkova, N. Steele, R. Neruda, M. Karny (eds.), Artificial Neural Networks and Genetic Algorithms. Springer, Wien (Computer Science Series) (2001) 324–327
- [16] Parsopoulos, K.E., Laskari, E.C., Vrahatis, M.N.: Solving ℓ_1 Norm Errors-In-Variables Problems Using Particle Swarm Optimizer. M.H. Hamza (ed.), Artificial Intelligence and Applications. IASTED/ACTA Press (2001) 185–190
- [17] Parsopoulos, K.E., Vrahatis, M.N.: Initializing the Particle Swarm Optimizer Using the Nonlinear Simplex Method. A. Grmela, N.E. Mastorakis (eds.), Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation. WSEAS Press (2002) 216–221

- [18] Press, W.H., Vetterling, W.T., Teukolsky, S.A., Flannery, B.P.: *Numerical Recipes in Fortran 77*. Cambridge University Press, Cambridge (1992)
- [19] Rao, S.S.: *Optimization: Theory and Applications*. Wiley Eastern Limited (1977)
- [20] Schwefel, H.-P.: *Evolution and Optimum Seeking*. Wiley (1995)
- [21] Shi, Y., Eberhart, R.C.: Parameter Selection in Particle Swarm Optimization. *Evolutionary Programming VII* (1998) 591–600
- [22] Yang, J.-M., Chen, Y.-P., Horng, J.-T., Kao, C.-Y.: Applying Family Competition to Evolution Strategies for Constrained Optimization. *Lecture Notes in Computer Science*, Vol. 1213. Springer-Verlag, Berlin Heidelberg New York (1997) 201–211

Various Approaches in Modeling of Algae Population

Jozef CHVAL, Martin PALKO
Department of Cybernetics and Artificial Intelligence,
Technical University of Košice
{Jozef.Chval, Martin.Palko}@tuke.sk

Abstract. A simulator of the ecologically important alga *Chlorella kessleri* can be very useful for biologists doing their experiments. To create it, we use the ALife methods. In the paper two approaches to the model design of the alga population are compared: multiagent and L-systems. The results from simulations show the differences of that two models.

Keywords: *ALife, biological modeling, Chlorella kessleri, L-system, multiagent, simulation*

1. Introduction

This paper is a contribution to the algae simulator project. The aim of the project is to create a simulator of a green algae specie to help the biologists to save financial and time expenses with some of their experiments. In this paper a new approach to the model is discussed. First the 2-dimensional multi-agent model was designed. Now we use L-systems to represent the alga and its environment, and compare the results to the agent model.

The simulated alga in our project is the *Chlorella kessleri*. This alga is one of the organisms that are able to adsorb and absorb various toxic elements (e.g. heavy metals) from the environment. Modeling of this biosorption process is particularly useful to predict its performance under different conditions. Computer simulations can then replace numerous tedious and expensive experiments [1].

2. Two approaches

Both our approaches are based on the ‘ALife’ methods. The essence is in building the model synthetically – define the basic elements and their interactions, and then – during the simulation – observe the emergent behavior of the whole [6].

2.1 Multi-agent model

The first model consists of two main parts: the alga cell and the environment (widely described in [2] and [3]). The cell is considered as a reactive agent. Its actions are determined by few rules, which implement the key phenomena concerning the lifecycle and the sorption process of the cell. The environment is a 2-dimensional discrete lattice consisting of three ‘substance’ layers and one ‘cell’ layer. The substance layers store the concentrations of the toxic substance, minerals, and metabolite over the environment. The cell layer serves to store the positions of the cells. In addition, the behavior of the whole model depends on values of about 30 parameters.

2.2 L-systems

The second model is based on L-systems. L-systems were introduced and firstly used by Aristid Lindenmayer in 1968 as a modification of formal grammars for the purposes of modeling of simple multicellular organisms development [4]. L-systems and their modifications became strong tool for generating fractals, self-similar structures, FASS (space-Filling, self-Avoiding, Simple and self-Similar) curves, modeling of development of biological structures, etc. As mentioned above, L-systems are modification of formal grammars. The essential difference between Chomsky grammars and L-systems lies in the method of applying the productions. In Chomsky grammars, productions are applied sequentially, whereas in L-systems they are applied in parallel and simultaneously replace all letters in a given word. In addition, in L-systems, there is no distinction between terminal and non-terminal symbols and every symbol has just one production (except some special types of L-systems). Basic classes of L-systems are: context-free (0L-systems), stochastic, context-sensitive and parametric L-systems [5].

2.3 L-systems based approach

In this paper we present a new approach in the field of L-systems based biological systems modeling. We use parametric stochastic context-sensitive L-system as a tool for representation of the algae cells and their environment, as well as interactions between the cells and environment, and among cells themselves. Because of L-system the environment was reduced from the 2-dimensional discrete lattice to the 1-dimensional string.

Each element of the environment is represented by one parametric symbol in the string. First two parameters of that symbol contain information about the environment. We decided to store the concentrations of the toxic substance and minerals. The concentration of the metabolite was excluded to make the model simpler, since it turned out not to be necessary at this level of experiments. The rest of parameters store information about alga (whether there is an alga cell in that element of the environment and what is the state of that alga). Key interactions between the algae and the environment, and among the algae themselves are represented by a set of L-system rules:

3. Experiments

Several simulations with various initial conditions were run to find out the differences and the common features of the two models. The operation of the models was observed via the statistical data: the population growth, and the metal sorption dynamics of that population.

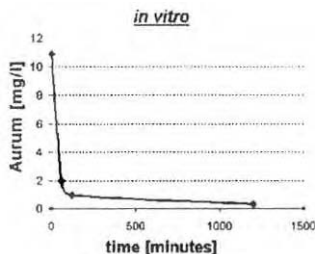


Figure 1. A typical result of a real experiment. (Sorption of gold by algae biomass.)

3.1 Initial Conditions

The multiagent model was realized using the ObjectiveC programming language and the SWARM libraries [7], which allowed us to create the environment of relatively large size. For computer memory reasons, we defined the lattice of 100 x 100 elements. However, for the L-system model implementation, we had a rather slow simulation tool. Therefore we decided to use smaller measures (a 512-element string) for the environment with the L-system model. The initial number of algae cells was stated as a percentage of the element count to make the simulations comparable. The initial amounts of the toxic metal and minerals were equivalent.

3.2 Simulation Results

In the figures 2 – 5, there are graphs with the results of several computer simulations using both multiagent ('a' column) and L-systems ('b' column) approaches. The algae population growth in the environment without the toxic metal is represented in the figure 2. The subsequent figures display the population dynamics, as well as the corresponding metal sorption rate, in the case the environment is polluted by the toxic metal.

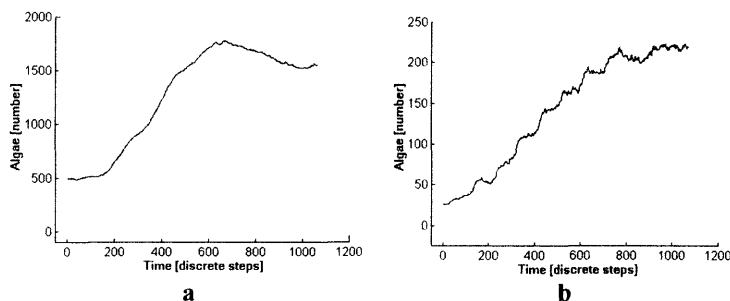


Figure 2. The population growth in the environment without toxic metal. (The initial population size is 5%.)

4. Result Analysis

First, the steepness of the population dynamics. The growth and fall of the cell number using the multiagent model is apparently steeper than with the L-system. The reason is that the population growth depends on the ambient light intensity. In both models the intensity is determined by the number of the neighboring cells – the cells shadow each other. In the 2-dimensional multiagent model the light is influenced by 8 neighbor positions, while in the 1-dimensional L-system string there are only 2 neighbor positions considered.

The metal concentration decrease is similarly steeper in the multiagent simulations. It is possibly caused by faster population growth mentioned above, together with an additional dimension for moving in this model.

The graphs in the 'a' column are smoother than the 'b' graphs, because the amounts of cells in the L-system simulations are about 20 times smaller, so the perturbations are more apparent.

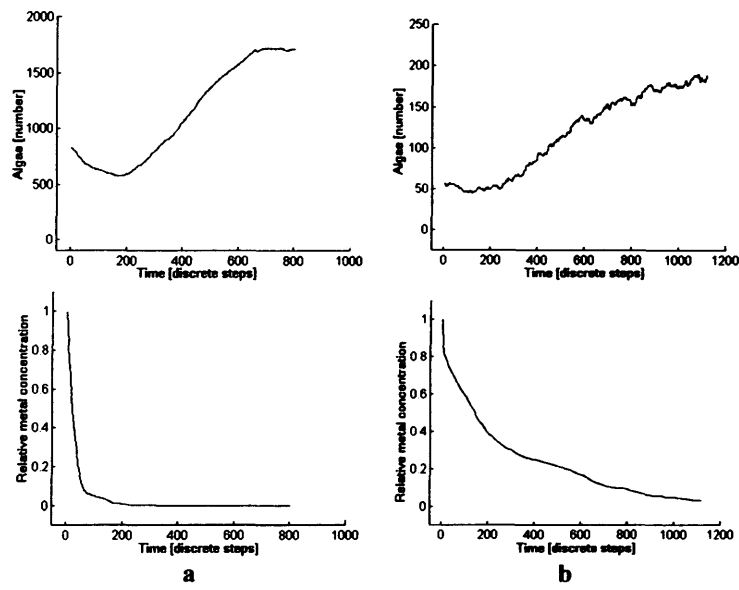


Figure 3. Simulation with toxic metal and small initial number of algae cells (10% of the environment elements were occupied).

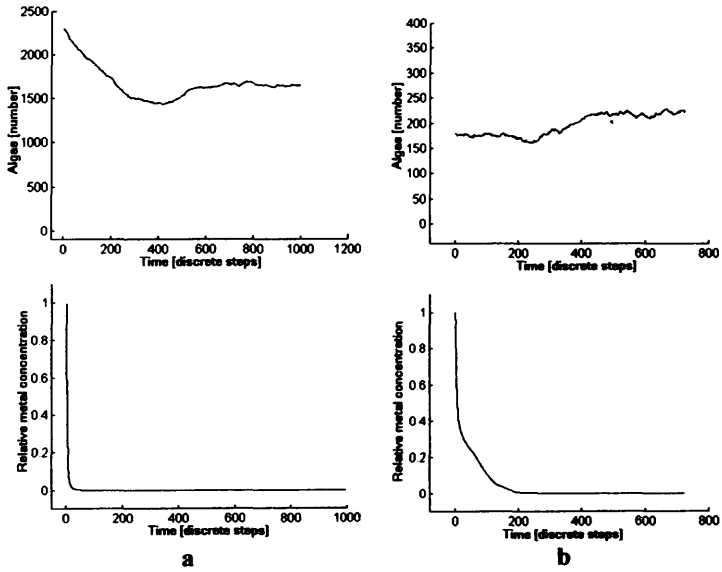


Figure 4. Simulation with toxic metal and medium initial number of algae cells (25% for the a-simulation and 35% for the b-simulation).

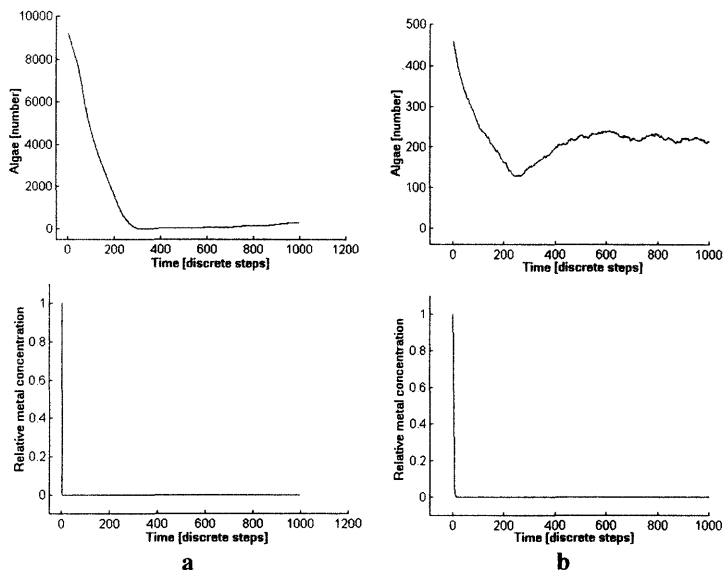


Figure 5. Simulation with toxic metal and large initial number of algae cells (90% of the environment elements were occupied).

In the experiment with the medium amount of the algae mass (figure 4), we can observe that the cell number tends to get stable. In the case of multiagent simulation it is about 16 % of the environment size, and in the case of L-system it is about 45 %. The reason is the same as in the first paragraph – the light and the shadow.

And finally, if we compare the real metal (e.g. gold) sorption dynamics (figure 1) to the artificial ones (figures 3 – 5), we can see that all the simulations roughly follow the real experiment. The deviations of the speed of the sorption process are caused just by the various initial population sizes.

5. Conclusion

The result analysis describes several differences between two models that were realized. We think, some of them can be reduced (even eliminated) adjusting the L-systems model, and it can be sufficient at a certain level of experiments. (The simulations follow the real sorption dynamics quite well.) But these results also encourage us to create and test a 3-dimensional model (possibly using the multiagent approach).

Acknowledgement

This work is supported by grant VEGA 1/8135/01 – “Life Simulators and their applications”.

References

- [1] Volesky, B.: Detoxification of metal-bearing effluents: biosorption for the next century. In: Hydrometallurgy, vol. 59 (2001), Elsevier, pp. 203-216.
- [2] Polak, M., Palko, M., Mihalo, M.: Modeling of the Sorption Mechanisms of the Green Algae Using the Artificial Life Simulators. (in Slovak) In: Cognition and Artificial Life. (in Czech) FPF SU Opava, 2001, 225-236.
- [3] Kadukova, J., Mihalo, M., Palko, M., Polak, M., Vojtun, J.: Algae Simulator Tuning by Means of Evolutionary Algorithms. In: Matousek, R., Osmera, P.: Mendel 2001 (7th International Conference on Soft Computing), Kuncik, Brno, 2001, pp. 111-115. ISBN 80-214-1894-X.
- [4] Lindenmayer, A.: Mathematical models for cellular interaction in development I,II. In: Journal of Theoretical biology, vol. 18 (1968), pp. 280-315.
- [5] Lindenmayer, A.: The Algorithmic Beauty of Plants. Springer-Verlag, New York 1990.
- [6] Csonto, J.: Artificial Life (in Czech). In: Artificial Intelligence 3. (Marik, V. ed.), Academia, Praha, 2001.
- [7] Swarm Development Group: <http://swarm.org>.

Estimation Distribution Algorithm for Mixed Continuous-discrete Optimization Problems

Jiří OČENÁŠEK, Josef SCHWARZ

Brno University of Technology, Faculty of Information Technology,
Božetěchova 2, 612 66 Brno, Czech Republic
ocenasek@fit.vutbr.cz, schwarz@fit.vutbr.cz

Abstract. In recent few years expressive progress in the theory and practice of Estimation of Distribution Algorithms (EDA) [1] has appeared, where the classical genetic recombination operators are replaced by probability estimation and stochastic sampling techniques. In this paper we identify some disadvantages of present probabilistic models used in EDAs and propose more general and efficient model for continuous optimization problems based on the decision trees. The new variant of EDA is capable to solve mixed continuous-discrete optimization problems.

Keywords: Estimation Distribution Algorithm, Bayesian Optimization Algorithm, Bayesian network, Gaussian network, decision tree, CART model

1. Introduction

For the discrete optimization problems the Bayesian network [1] and its decision graph refinement [2] are used in Estimation Distribution Algorithms as the general models to encode the parameter dependencies.

For continuous domains there are several approaches used to discover and encode the parameter dependencies – binary encoding [3,4], Gaussian networks [5], Gaussian kernels and mixtures of Gaussians [6].

In the following chapter we show that all present models used for optimization of continuous functions are either inaccurate or do not allow proper mixing of building blocks. In chapter 3 we proposed the usage of decision trees to eliminate the disadvantages of present models. In our Mixed Bayesian Optimization Algorithm (MBOA) we implemented the trees with mixed decision nodes, so MBOA is suitable for both continuous and/or discrete optimization problems.

2. Present methods and approaches

In EDA individuals in the population are treated as vectors of instantiations of n random variables X_i , each random variable represents one parameter of a solution

$$X = (X_1, X_2, \dots, X_n) \quad (1)$$

The goal of optimization is to find an optimal solution X^* from domain D such that

$$X^* = \arg \max \text{fitness}(X), \quad \forall X \in D \quad (2)$$

In case of discrete combinatorial problem all variables X_i are discrete. Each generation the space of possible solutions is effectively sampled using the Bayesian network (BN) [1], which encodes the conditional probabilities

$$p(X) = \prod_{i=1}^n p(X_i | Pa(X_i)) \quad (3)$$

The Bayesian network is constructed in the incremental way - for each variable X_i a large number of statistical tests is performed to form the set of "parent" variables $Pa(X_i)$ which significantly affect the value of variable X_i .

In the case of a continuous optimization problem it is possible to transform each continuous parameter into binary parameters and use the Bayesian network again. However, with the increasing precision of encoding the complexity of BN grows exponentially. The other way of solving continuous optimization problem by EDA is to let the parameters X_i be continuous (like in evolutionary strategies) and find a proper model for this continuous domain.

The basic model is the Gaussian network [5], where the mean value m_i of each random variable X_i is affected by linear combination of "parent" values X_j . This simple regression model is able to describe the fitness function with only one local extreme (the shape of reliable sampling region is elliptical).

Instead of using one probability density function (PDF) it is possible to use each solution as a source of elementary normal PDF - the Gaussian kernel [6]. It seems this may improve the sampling accuracy, but in fact no linkage information necessary for proper mixing of building blocks is provided. The Iterated Density Estimation Evolutionary Algorithm (IDEA) [6] uses clustering strategy to divide the samples into linear clusters and for each cluster one Gaussian network is used. This is more general than usage of the single linear regression for the entire space of parameters, but no mixing of building blocks between clusters is possible.

3. MBOA algorithm

In our approach we propose and implement the EDA algorithm based on the decision trees. Binary decision trees have been successfully used in EDA [2] for discrete optimization problems - with the Bayesian network. In our work we have focused especially on non-binary problems.

3.1 CART model

From the area of data mining we adopted the idea of CART model [7] (Classification and Regression Tree), which is usable also for continuous and categorical splits. For each "target" random variable X_i we build one decision tree. The split nodes of i -th decision tree are used to cut the domain of parent variables $Pa(X_i)$ into parts, where the variable X_i is more linear or predictable. We start building the decision trees from empty trees and we recursively add the splitting nodes until no splitting is favourable.

```

Function RecursiveSplitting(Population Pop,
    TargetVariable  $X_i$ ,
    ListOfCandidateSplitVariables Pa)
    : DecisionTreeNode
Begin
     $f_{Temp} := \text{EstimateElementaryPDF}(\text{Pop}, X_i, \text{Pa})$ ;
    If "model is too detailed" then
        return new LeafNode( $f_{Temp}$ );
    For Each Variable  $X_j$  in Pa do
         $E_j := \text{Find\_optimal\_split\_on\_}X_j\_\text{with\_respect\_to\_}X_i$ ;
         $\text{ModelGain} := \text{Evaluate\_the\_split\_gain}("X_j \leq E_j", X_i)$ ;
        Save the  $X_j$  and  $E_j$  with the highest ModelGain;
    End for
    Pop1 := SelectIndividuals (Pop, " $X_j \leq E_j$ ");
    Pop2 := SelectIndividuals (Pop, " $X_j > E_j$ ");
    return new SplitNode(new SplitCondition(" $X_j \leq E_j$ "),
        RecursiveSplitting(Pop1,  $X_i$ ,  $\text{Pa} \setminus \{X_j\}$ ),
        RecursiveSplitting(Pop2,  $X_i$ ,  $\text{Pa} \setminus \{X_j\}$ ));
End;

```

The leaf nodes define the elementary models for obtaining the “target” variable X_i . For continuous variables one-dimensional normal PDF is estimated and used as the leaf. We are also able to use Gaussian kernels or the linear regression model known from Gaussian network.

3.2 Metrics for model construction

In the RecursiveSplitting procedure the list of candidate split variables is given in advance, which allows the future parallelization, see [8]. The step of split condition finding and evaluation is essential. In the present version we use more sophisticated metrics to find the optimal split boundary E_j^* :

$$E_j^* = \arg \max_{\forall E_j \in X_i} \text{Gain}(E_j) \quad (4)$$

$$\text{Gain}(E_j) = \max_{\forall E_j \in X_i} \frac{\sum_{r \in \{\text{left}, \text{right}\}} \sum_{s \in \{\text{up}, \text{down}\}} \Gamma(n_{k,j} + 1) \cdot \Gamma(\sum_{r \in \{\text{left}, \text{right}\}} \sum_{s \in \{\text{up}, \text{down}\}} (n_{k,j} + 1))}{\sum_{r \in \{\text{up}, \text{down}\}} \Gamma(\sum_{s \in \{\text{left}, \text{right}\}} (n_{k,j} + 1)) \cdot \sum_{r \in \{\text{left}, \text{right}\}} \Gamma(\sum_{s \in \{\text{up}, \text{down}\}} (n_{k,j} + 1))} \quad (5)$$

We derived the equation (5) from the Bayes-Dirichlet metrics (BDe, see [9]), meaning of symbols $n_{\text{left}, \text{down}}$, $n_{\text{left}, \text{up}}$, $n_{\text{right}, \text{down}}$, $n_{\text{right}, \text{up}}$ is shown in Fig. 1.

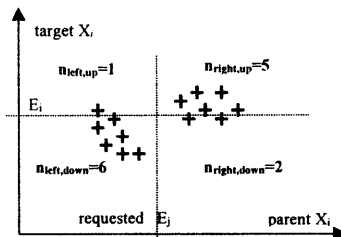


Figure 1. All individuals in the population are classified into four groups according to conditions “ $X_j \leq E_j$ ” and “ $X_i \leq E_i$ ”. The symbols $n_{\text{left}, \text{down}}$, $n_{\text{left}, \text{up}}$, $n_{\text{right}, \text{down}}$, $n_{\text{right}, \text{up}}$ represent number of individuals in each quadrant.

The next advantage of decision trees is the backward-compatibility with discrete domains. For binary split nodes the split condition is straightforward “ $X_j \leq 0$ ”, so we can apply the equation (5) for $E_j=0$. Each leaf for binary target determines the frequency of occurrence of value 1 in the part of population, which fulfills the split-conditions corresponding to the path from the root.

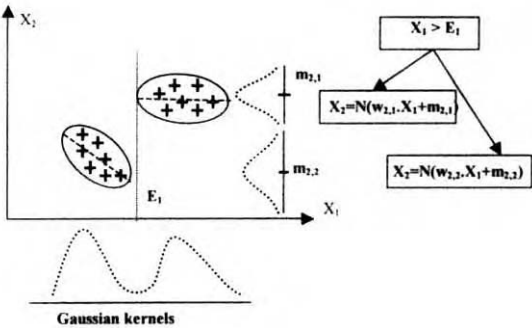


Figure 2. The population precisely approximated by decision trees.

We are also able to optimize non-binary discrete problems in their natural alphabetical encoding. In this case we use hillclimbing algorithm to split the set of possible X_j values into left and right subset. The variable s in (5) goes through all possible values of X_i instead of only two categories {up,down}.

The next advantage of MBOA is the utilization of RTR (Restricted Tournament Replacement) proposed by Martin Pelikan in [10]. In the replacement stage each new individuals competes with the Euclidean-nearest individual among 20 randomly selected individuals. This ensures better niching. As the next improvement we utilized the fitness value in the same way as the continuous parent variable - the population can be split according to certain fitness value into more-homogenous regions to achieve better divergence and model accuracy.

4. Experimental results

In the experimental part we compared the efficiency of our approach with the other mentioned EDA algorithms. First of all we proved the backward-compatibility with the binary domain. We compared the performance of MBOA with the Pelikan’s original simple BOA algorithm on several deceptive functions.

Algorithm/Benchmark	deceptive ₃ function, n=90	trap ₅ function, n=90
MBOA	43530 evaluations, N=3800	53280 evaluations, N=5500
Simple BOA	43510 evaluations, N=3800	54310 evaluations, N=5500

Table 1. Results for deceptive₃ and trap₅ function (see [2])- the average number of fitness evaluations required to reach the global optima in 10 runs, with min. population size N.

Secondly, we used some continuous benchmarks to compare MBOA with the best IDEA algorithms in [6]. IDEA1 uses non-clustered normal mixture of 10 PDFs, IDEA2 and IDEA5 use Euclidean 2-means clustering with normal mixtures of 10 PDFs, IDEA3 uses Mahalanobis leader 3½ clustering with normal mixtures of 10 PDFs, IDEA4 uses non-clustered normal mixture of 5 PDFs and IDEA6 uses Mahalanobis leader 5 clustering with normal mixtures of 5 PDFs.

Algorithm	Fitness evaluations to get the global optima (avg. for 10 runs, 7-digit precision)
MBOA with univariate Gaussian leafs	19790 , N=120
IDEA1	51832, N=1100
IDEA2	72552, N=1750
IDEA3	89718, N=2250

Table 2. Griewank function, problem size $n=5$, domain= $[-5,5]^5$.

Algorithm	Fitness evaluations to get the global optima (avg. for 10 runs, 7-digit precision)
MBOA with univariate Gaussian leafs	7690 , N=120
IDEA4	28903, N=1300
IDEA5	16596, N=750
IDEA6	22118, N=1000

Table 3. Michalewicz function, $n=5$, domain= $(0,\pi)^5$

Thirdly, we tested the scalability of MBOA on continuous two-deceptive function, see [3],[4] and we examined how the number of fitness evaluations grows with the problem size.

Algorithm	Two-deceptive function [3], 5-digit precision
MBOA with Gaussian-kernels leafs	$O(n^{1.75})$ ($n=10-50$, $N=300-3500$)
Simple BOA with fixed-width histogram discretization, see [3]	$O(n^{2.1})$ (4 bins, $n=10-40$, $N=1000-7400$)
Simple BOA with fixed-height histogram discretization, see [3]	$O(n^{2.1})$ (16 bins, $n=10-30$, $N=3500-17000$)

Table 4. The scalability for continuous two-deceptive benchmark

Fourthly, we prepared our own mixed continuous-discrete benchmark, where the chromosome of length $n=2l$ is composed of binary genes b_i and continuous genes c_i . This benchmark is very sensitive to correctness of dependency metric – mixed dependencies have to be discovered to find the global optimum.

$$f_{mixed}(b_0, c_0, b_1, c_1, \dots, b_{l-1}, c_{l-1}) = \sum_{i=0}^{l-1} f_m(b_i, c_i) \quad (6)$$

$$f_m(b_i, c_i) = \begin{cases} 0.8 - 0.5 * c_i & b_i = 0 \\ 0.8 - c_i & b_i = 1 \wedge c_i \leq 0.8 \\ 5 * (c_i - 0.8) & b_i = 1 \wedge c_i > 0.8 \end{cases} \quad (7)$$

Problem size	n=10	n=20	n=30	n=40	n=50
MBOA with Gaussian-kernels leafs	23100 N=600	67800 N=1200	126000 N=1800	214800 N=2400	291000 N=3000

Table 5. The average number of evaluation for mixed benchmark

5. Conclusion and future work

In this paper we have identified the limitations of present probability models [5],[6] and proposed more general and efficient model for continuous optimization problems based on the decision trees. Decision trees are more general model than Gaussian networks and more useful for building blocks evolution than mixtures of Gaussians. Moreover, our approach is compatible with discrete domains, so we introduce the new kind of EDA uniquely capable to solve mixed continuous-discrete optimization problems.

The empirical results show that in the field of discrete optimization problems our MBOA is backward-compatible with the original BOA algorithm and provides similar results. The results for continuous Griewank and Michalewicz function confirm our assumption that MBOA overcomes the IDEA approach [6].

On the two-deceptive continuous benchmark we showed that the scalability of MBOA $O(n^{1.79})$ is better than the scalability $O(n^{2.1})$ of discretization approach [3] based on histogram models. The MBOA was also able to solve our mixed continuous-discrete hard benchmark. The results indicate that the main potential of our algorithm lies in solving high-dimensional problems with stronger parameter nonlinearities.

This research has been carried out under the financial support of the Research intention no. CEZ: J22/98: 262200012 - "Research in information and control systems" (Ministry of Education, CZ) and the research grant GA 102/02/0503 "Parallel system performance prediction and tuning" (Grant Agency of Czech Republic).

References

- [1] Pelikan, M., Goldberg, D. E., Lobo, F.: A Survey of Optimization by Building and Using Probabilistic Models. IlliGAL Report No.99018, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, September 1999.
- [2] Pelikan, M., Goldberg, D. E., Sastry, K.: Bayesian Optimization Algorithm, Decision Graphs, and Bayesian Networks. IlliGAL Report No. 2000020, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, May 2000.
- [3] Pelikan, M., Goldberg, D. E., Tsutsui, S.: Combining the Strengths of the Bayesian Optimization Algorithm and Adaptive Evolution Strategies. IlliGAL Report No. 2001023, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, June 2001.
- [4] Tsutsui, S., Pelikan, M., Goldberg, D. E.: Evolutionary Algorithm Using Marginal Histogram Models in Continuous Domain. IlliGAL Report No. 2001019, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, March 2001.
- [5] Larrañaga, P., Etxeberria, R., Lozano, J. A., Peña, J. M.: Optimization by learning and simulation of Bayesian and Gaussian networks. Technical Report EHU-KZAA-IK-4/99, University of the Basque Country, December 1999.
- [6] Bosman, P. A. N., Thierens, D.: Mixed IDEAs, Utrecht University technical report UU-CS-2000-45, December 2000.
- [7] Chipman, H., George, E., McCulloch, R.: Bayesian CART model search, Journal Am. Statist. Assoc., 93, pp. 937-960.
- [8] Očenášek, J., Schwarz, J.: The Parallel Bayesian Optimization Algorithm, Proceedings of the European Symposium on Computational Intelligence, Physica-Verlag, Košice, Slovak Republic, September 2000, pp. 61-67.
- [9] Heckerman, D., Geiger, D., Chickering, M.: Learning Bayesian networks: The combination of knowledge and statistical data. Technical Report MSR-TR-94-09, Microsoft Research, Redmond, WA, 1994
- [10] Pelikan, M., Goldberg, D. E.: Escaping Hierarchical Traps with Competent Genetic Algorithms. IlliGAL Report No. 2001003, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, January 2001.

On Dimensionality Reduction of High Dimensional Data Sets

CHIZI B., SHMILOVICI* A., MAIMON O.

*Department of Industrial Engineering, Tel – Aviv University
POB 39040, Ramat Aviv, Tel Aviv 69978, Israel*

**Department of Industrial Engineering and Management
Ben-Gurion University of the Negev Beer-Sheva 84105, Israel*

Abstract. High dimensional databases are demanding in terms of the computational power required for their processing. Dimensionality reduction can effectively reduce the costs of various operations (e.g. classification). This research presents an explanation why dimensionality reduction is often possible with minimum information loss. Three kinds of greedy dimensionality reduction techniques are presented: Information Gain (Entropy), Polytomous Logistic Regression and random removal of attributes. An empirical comparison of the effect of the above methods on 10 benchmark data-sets revealed that a relatively simple logistic regression method provided mostly the best results

Keywords: Dimensionality Reduction, Data Mining, Logistic Regression

1. Introduction and Problem Definition

Data-mining operations generally search for meaningful patterns in raw data-sets. In a classification operation - for example - we seek for rules that can predict the partitioning of the data into different classes having different physical meanings.

Data-mining operations are computationally intensive: Fig. 1 describes the typical tradeoff between the error rate of a classification model and the cost of using the model. The cost is a measure of complexity of the classifier, the time required for the algorithm to run, and the size of the data set. Since the first two factors are intimately related to the size of the data set, in this work we will only relate the cost to the size of the data set.

Theoretically, if we know the exact functional relation between the cost and the classification error, we might be tempted to look for the ideal classifier - that which produces the minimum error rate ϵ^* based on the cost of all the data h^* . Other times, we might prefer to use an inferior classifier h that uses only a part of the data $h \leq h^*$. In practice, the tradeoff curve of Fig. 1 is seldom known, and generating it might be computationally prohibitive.

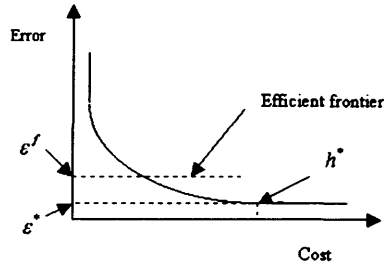


Figure 1: Typical cost-error relation in a classification operation

Since we do need a solution, we would like to identify the smallest cost of a data-set (with minimum size for h), that can keep the error rate below ε^f . Motivated by practical considerations, the purpose of this work is to investigate the power of simple greedy approaches for the reduction of the data-sets. The simplest reduction is the removal of insignificant attributes. This work will investigate the power of simple greedy approaches to attribute removal.

Not only is it a costly process, but it also contradicts our initial assumption that all the information (attributes) is required in order to achieve maximum accuracy, and while some attributes are less important, no attributes are irrelevant or redundant. We will not consider feature selection approaches for dimensionality reduction.

2. Used Methods and Approaches

Using geometrical terms, when we are reducing dimensionality, we are losing some volume which might contain information about the domain. Jimenez and Landgrebe (1998) present a geometrical analysis of high-dimensionality spaces. They conclude that most of the high-dimensionality space is empty and by judicious selection of the dimension that will be reduced, we will not lose significant information. In the following lemma, we demonstrate in a simpler way the notion that most of the high dimensional space is empty:

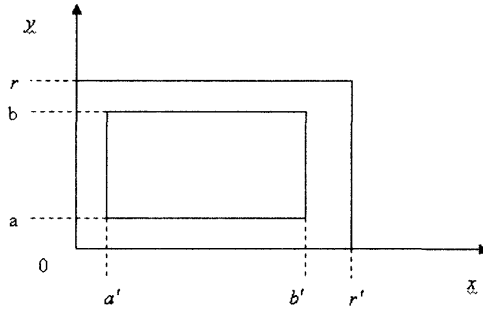
Lemma 1

Consider the space that belongs to a cube of length r : $U \in [-r, r]^d$ in high dimensional space $d \gg 1$. Then most of the volume outside a thin shell of thickness $\delta \ll r$ is empty. That is:

$$[-r, r]^d - [-r + \delta, r - \delta]^d \rightarrow [0]$$

Proof:

Consider a slice of that hypercube ($[-r, r]^d$) for $d=2$ like the one presented in Fig. 2, where $a = a' = \delta, b = b' = r - \delta$:

Figure 2: – A slice of hypercube for $d = 2$

If we assume uniform distribution of the data, or we completely ignore the distribution of the data, then the probability of the data to be found on the shell is each single dimension x , y is proportional to the shell thickness:

$$p(x) = p(0 \leq x \leq a' \cup b' \leq x \leq r')$$

$$p(y) = p(0 \leq y \leq a \cup b \leq y \leq r)$$

In two dimensions, the probability of the data not to be found on the two dimensional shell is:

$$p(\bar{x}, \bar{y}) = (1 - p(x))(1 - p(y))$$

And for $d = n$ dimensions, the probability of the data not to be found on the n dimensional shell is $p(\bar{x}, \bar{y}) = [(1 - p(x))(1 - p(y))]^n$.

As $n \rightarrow \infty$ $p(\bar{x}, \bar{y}) \rightarrow 0$. Thus, we can conclude that in high dimensional space, most of the volume of the data is concentrated on the multi-dimensional shell, and the probability to find it inside the shell is close to 0. Q.E.D.

In practice, the data is not evenly distributed among all the sides of the shell. Elimination of one of the dimensions (as in Fig. 2) can result in lack of discrimination regarding which side of the shell the data came from. Our greedy algorithms will search for the possibility that the data is concentrated in only one side of the shell, and eliminate that dimension.

As stated above we used the following approaches for dimension reduction:

1. The random approach: We will randomly pick an attribute and reduce it from the data set. For n attributes, the probability function for picking the best candidate for reduction is distributed uniformly:
2. Using Entropy or Information Gain for reducing dimensionality: The conditional entropy of a random variable Y , given another random variable X is defined as $H(Y/X) = -\sum p(x,y) \log p(y/x)$. The mutual information or the information gain between two random variables X and Y (Mitchell, 1997) is defined as a decrease in entropy of Y as a result of knowing X (and vice versa), namely:

If X and Y are independent, then knowledge of X will have zero contribution to the entropy of Y . The same concept is extended (Cover, 1991) to the mutual information between a target attribute a_{target} and n other attributes (A_1, \dots, A_n) . Thus, if we detect that a candidate attribute has minimal contribution to the entropy of the target attribute, then it has negligible influence on the target attribute, and it is a good candidate for elimination from the data-set.

1. Reducing dimensionality with polytomous logistic regression: Consider the class variable a_{target} which can be one of K possible classes. It is assumed that the response variable a_{target} is a nominal scaled variable; that is, the classes have no natural ordering. However, the method can still be used when the response a_{target} is an ordinal-scaled variable. Let A denote the vector of attributes. Let us define the following regression model for a_{target} based on the attribute vector A :

$$\Pr(a_{target} = j|A) = \frac{e^{g_j(A)}}{1 + \sum_{m=1}^{K-1} e^{g_m(A)}}, \quad j = 1, 2, \dots, K-1 \quad (1)$$

$$\Pr(a_{target} = K|A) = \frac{1}{1 + \sum_{m=1}^{K-1} e^{g_m(A)}} \quad (2)$$

Where:

$$g_j(A) = \beta_{j0} + \sum_{k=1}^N \beta_{jk} a_k, \quad j = 1, 2, \dots, K-1 \quad (3)$$

The model is transformed to produce the generalized *logit* as follows:

$$\log \left[\frac{\Pr(a_{target} = j|A)}{\Pr(a_{target} = K|A)} \right] = g_j(A) = \beta_{j0} + \sum_{k=1}^N \beta_{jk} a_k, \quad j = 1, 2, \dots, K-1. \quad (4)$$

Lemma 2 provides an estimate for the growth of the error due to the elimination of one of the attributes from the attribute vector.

Lemma 2.

Deleting the attribute which has $\min \beta_{jk}$ from formula (4) will lead to a bounded error:

$$\Pr(a_{target} = j|A-1) = \frac{1}{2} + \frac{\varepsilon}{\sqrt{2}} \quad (5)$$

Proof:

First, consider a 2-dimensional attribute vector: $A \in (a_1, a_2)$. From (1):

$$\Pr(a_{target} = j|A) = \frac{e^{\beta_{j0} + \beta_{j1}a_1 + \beta_{j2}a_2}}{1 + e^{\beta_{j0} + \beta_{j1}a_1 + \beta_{j2}a_2}} \quad (6)$$

Define $\varepsilon = \beta_{j0} + \beta_{j1}a_1 + \beta_{j2}a_2$. Then by first order Taylor expansion for small ε :

$$\Pr(a_{target} = j|A) = \frac{e^\varepsilon}{1 + e^\varepsilon} \approx \frac{1 + \varepsilon}{2 + \varepsilon} \approx \frac{1 + \varepsilon}{2} = \frac{1}{2} + \frac{\varepsilon}{2} \quad (7)$$

Assuming the worst-case scenario, a_1 and a_2 are independent. By readjusting the regression coefficients, a_1 and a_2 can be approximated as standardized normal variables $N(0, 1)$. Then,

each variable has a maximal contribution of $\frac{1}{\sqrt{2}}$ to the variance of (3). After reducing one

dimension, ε will be changed and can be written as: $\bar{\varepsilon} = \frac{\varepsilon}{\sqrt{2}}$.

And (7) can be recalculated as: $\Pr(a_{\text{target}} = j | A - 1) \approx \frac{1}{2} + \frac{\varepsilon}{\sqrt{2}}$

The above argument can be easily repeated for $n+1$ attributes: In the worst case scenario, there are n variables approximated by standardized normalized variables, each variable has

a contribution of $\frac{1}{\sqrt{n}}$ to the total variance. And there is the contribution of the attribute

with $\min \beta_{jk}$ which will have the biggest variance contribution to (4). Elimination of that variable will result in:

From Lemma 2 we can conclude that the bound will be tighter when n is large (i.e. dealing with high dimensional data).

3. Experiments and Results

The 10 benchmark data sets from the UCI Machine Learning Repository (Blake and Merz, 1998) were used to compare the effect of the three simple greed algorithms for the above dimensionality reduction. The C4.5 classification algorithm (Mitchell, T. M. 1997) with 10 fold cross validation was used to generate the classification accuracy, and variance estimates. The results are presented in table 1. Column 2 presents the number of attributes in each database, column 3 presents the classification accuracy estimates without attribute removal, and columns 4,5,6 present the classification accuracy estimates for each one of the algorithms in section 2, resulting from the removal of just one attribute.

Data set	Number of attributes	C5 - Accuracy	Entropy - Accuracy	Logit - Accuracy	Random - Accuracy
Iris	4	94 ± 1.8	95.3 ± 1.4	95.3 ± 1.7	95.3 ± 1.7
Liver	6	68.1 ± 2.4	66.7 ± 3.0	57.9 ± 3.1	57.8 ± 3.1
Diabetes	8	75.3 ± 1.9	73.7 ± 1.1	75.4 ± 1.8	69.4 ± 1.6
Glass	9	70.6 ± 2.8	67.7 ± 2.6	70.7 ± 3.2	67.7 ± 2.6
Breast	10	95 ± 1.0	93.8 ± 0.7	94.5 ± 1.2	93.8 ± 0.7
Wine	13	91.6 ± 1.2	93.8 ± 1.5	93.8 ± 1.5	93.8 ± 1.5
Credit	14	86.4 ± 1.3	84.2 ± 1.2	85.8 ± 2.0	84.2 ± 1.2
Heart	14	78.8 ± 1.1	78.8 ± 1.1	78.8 ± 1.1	77.8 ± 1.2
Chess	36	99.6 ± 0.1	98.9 ± 0.1	99.5 ± 0.1	98.9 ± 0.1
Lung	56	40 ± 6.2	34.2 ± 5.3	53.3 ± 7.9	40.8 ± 8.5

Table 1. Classification error estimates before and after 1 attribute removal

4. Results Analysis

From Table 1 we can see that Logit managed to reduce dimensionality with minimum damage to accuracy 6 times. The Information Gain managed to reduce dimensionality with

minimum damage to accuracy only once. The Random approach did not manage to do better than the other approaches, it even did worse.

From the above results we can conclude that there are two obvious alternative candidates for dimension reduction: Logit and Information Gain. The logit based method managed to perform slightly better than the information gain based method.

We used the ratio test to estimate which approach is better on high dimensional data-sets: The linear correlation coefficient was estimated between the ratio between the errors of the two approaches (Logit and Information Gain) and natural log of number of attributes of each data set.

The estimated linear correlation coefficient is -0.7624. This result is quite encouraging as it evidently indicates that we should use "Logit" for high dimensionality data sets.

5. Conclusion

Most of the potential applications of Computational Intelligence for real-world problems involved high dimensional data sets. One of the future trends in Computational Intelligence is to enable known algorithms and methodologies to operate on real-world, high dimensional datasets. Dimension Reduction is one of the important ways to deal with this problem.

This work investigated the effect of dimensionality reduction on high dimensional data-sets. It was demonstrated that in high dimensional spaces, even simple greedy algorithms for dimensionality reduction may not necessarily lead to a significant reduction in classification accuracy. Experiments with three greedy algorithms for dimensionality reduction suggest that the relatively simple polytomous logistic regression can produce results which are better than the more standard "Information Gain" approach.

References

- [1] Blake, C.J. & Merz, C.J. (1998). UCI Repository of machine learning databases.
- [2] Cover, T. M.(1991). *Elements of Information Theory*. Wiley.
- [3] Jimenez, L. O. (1998). Supervised Classification in High- Dimensional Space: Geometrical, Statistical, and Asymptotical Properties of Multivariate Data. *IEEE transactions on systems, man, and cybernetics*. Vol. 28, no. 1.
- [4] Mitchell, T. M. (1997). *Machine Learning*. McGraw-Hill.

Neocognitron and its applications in image processing

M. Šamulka^{1,2}, P. Kostelník^{1,2}, M. Hudec¹
CIT-Center for Intelligence Technologies,
Letná 9, 040 20 Košice, Slovak Republic
Faculty of Electrical Engineering and Informatics,
Technical University of Košice¹
&
Institute of Informatics, Slovak Academy of Sciences,
Bratislava, Slovak Republic²
samulka@neuron.tuke.sk

Abstract. Biological inspiration for neural technology has various degree of implementation and its application to brain-like systems is a very challenging problem. There are a number of different approaches to these neural technologies including ART-like neural networks, various Malsburgs models and the whole family of neocognitron systems based on Hubel and Wiesel model (H-W model) of the visual cortex and its information processing.

The presented work represents the experience with neocognitron technology developed by Prof. Fukushima in late 80-ties to identify the type of formations of the mobile robots on the soccer field. This information should be complementary to each soccer team and should represent an intelligent coach-like advisor for the teams to evaluate a type of situation on the playground. The input should be an image of the playground and the output should be a categorization to the types of formation of the team e.g. attacking formation, defending formation and so on. This information should be considered as additional for the selected team.

The biological inspiration here is very strong and presents a model of the visual system for real-time processing requirements. There will be briefly described general motivation and overall soccer system architecture. The main aim of this is to present experience with implementation and application of neocognitron neural network in the problem of estimation of the strategic distribution of soccer players on the playing field and its identification by the help of biologically inspired system.

Keywords: biologically inspired system, Neocognitron, neural networks, image processing, model of the visual cortex, mobile robots, hand-written letter-character recognition, multirobot control, soccer game.

1. Introduction

One of the general problems in multiagent systems design and implementation is other agent modeling especially when agents are fully autonomous physical embodied robots performing their decisions and actions in real-time noisy environments with high level of uncertainty. In such a dynamic and uncertain environments the complexity of other agents modeling arises very quickly and it is often too hard for each autonomous agent -robot to make decision depending on semantic context of the whole actual situation with all its features. This kind of situations often tends to arise of agent decision process complexity what usually causes delays and errors in action performing and communication overloads.

The main motivation is to reduce complexity of the simple agent decision process by providing additional information about types of actual situations on the soccer playground offered by coach-like system.

Implementation of neocognitron network as the main part of decision supporting coach-like system is based on experience with hand-written letter-character recognition. The situation - positions of the robots on the soccer playground - can be transformed into the special type of curve with variable design. After this transformation the problem of pattern recognition is similar to hand-written letter-character recognition and classification.

2. Present methods and approaches

The system architecture for multi-robot control in a soccer game can be considered as a combination of two teams of physical robots acting on the playground, vision camera overlooking the board connected to centralized interface computer, two multiagent systems connected as clients to central computer and two vision systems. Each multiagent system contains one team of agents as the "minds" of the robots on the playing field. Each vision system is related to one team and uses its own image processing methods. Figure 1. sketches the building blocks of the proposed architecture. The complete overall multirobotic system is modular, fully autonomous and consists of two main processing cycles:

- **Vision loop:** The task of the vision system is to perceive dynamic environment and to extract relevant information from sensed image (e.g. to extract positions and direction vectors of robots and the ball, to make the simulation of additional sonar-like sensory information for local view of each agent, to analyze actual situation on the playground, etc.). Perceived and processed visual information is transferred to the host computer where it is distributed to each client module (to each multiagent system and then to each agent).
- **Action loop:** Information from the visual system is processed by each agent and used as a base for what to do next ecision process. Next action of each agent is depending on the context of situation derived from agents increasing knowledge, actual agent's state and processed perceived information. Code of the new action is transferred to the related situated robots and executed on the playing field.

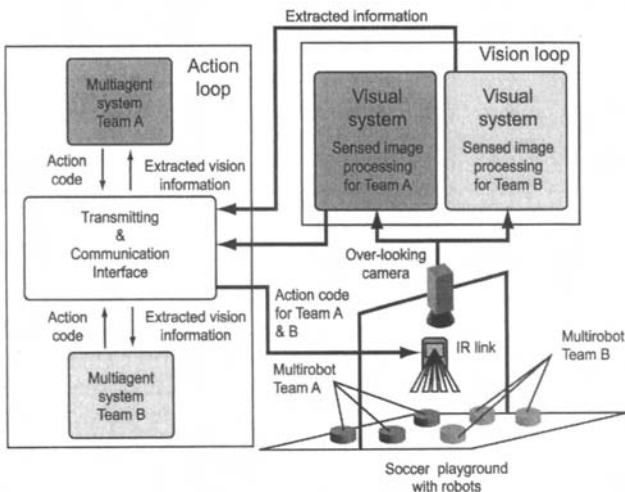


Figure 1. Building blocks of architecture for multirobot control.

Proposed architecture can be considered as modular where two independent teams can be connected to a central interface computer. Each team is enabled to use its own methods for image processing and multi-robot control.

Neocognitron network

The biologically inspired neocognitron network is made up of a hierarchy of processing 'stages' (see Figure 2.). Each stage comprises two different types of layers of cells:

- **the S-cell layer** (simple - feature detecting layer): All the cells in these layers have trainable weights except the inhibition one. Over the course of a training session they evolve to detect specific patterns of activation. The cells are grouped into cell-planes. All the cells in the same cell-plane detect the same feature but at different locations. Examples of the sort of features that the cells in the first stage might learn to recognize can be seen next to each cell-plane in the diagram. For S-cell detail see Figure 4.
- **the C-cell layer** (complex - blurring layer): all the cells in these layers have fixed weights. These layers essentially blur the pattern of activation coming from the previous feature detecting layer. There is a cell-plane in this layer for each cell-plane in the previous layer. Each cell-plane is only connected to the cell-plane directly below it. The activation of a single cell in a feature detecting cell-plane would produce a small area of activation in the corresponding blurring cell-plane. This means that the subsequent stage is trained on the approximate location of a specific feature. This makes the network tolerant to distorted images where the relative positions of features may vary slightly.

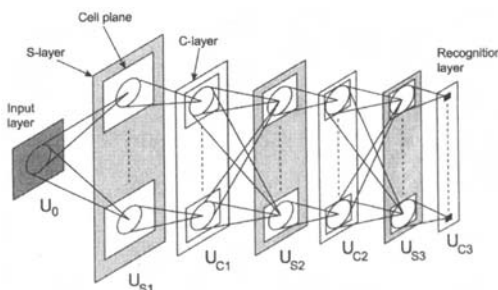


Figure 2. The neocognitron network hierarchic architecture.

The layers themselves are made of a number of parallel cell planes. Information flows up through the network, the output of one stage being the input to the next stage. The output of the network is one cell at the highest stage producing a high activation. This corresponds to the class that the network has assigned to the pattern being presented.

Formation type recognition – the description

- Image sensed using camera hanging above the soccer playground is processed by off-line learned multilayer perceptron and positions and direction vectors of the robots and the ball are extracted
- this information is used as input to Bezier's curves construction unit generating the patterns for each soccer team (see Figure 5.)

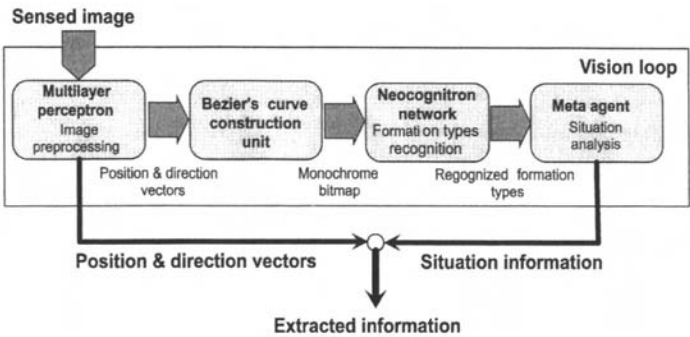


Figure 3. Building blocks of visual system for formation types recognition and situation analysis using biologically inspired neocognitron network.

- both patterns are processed by neocognitron network and classified to classes representing typical formations (see Figure 6.)
- recognized formations are processed by on-line learning meta agent and the strategic coach-like advice is derived from overlapping formation types and ball position

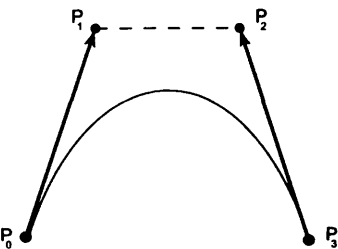


Figure 4. Example of Bezier's curve of 4-th order constructed from four points

Very often used type of Bezier's curve is Bezier's curve of 4-th order assigned by four points P_0 - P_3 . It is defined as:

where $t \in <0, 1>$ and B_0, B_1, B_2, B_3 are cubic multinomials:

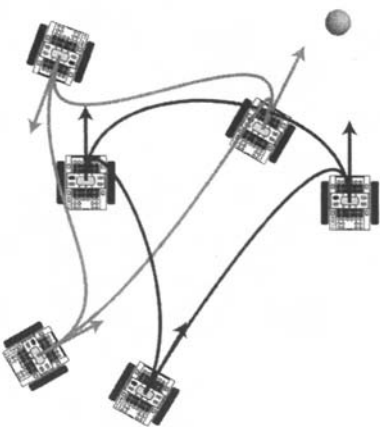


Figure 5. Example of Bezier's curves construction from positions and direction vectors of physical robots.

3. Experimental results

The basic feature of neocognitron network is its robustness against noise, rotation, displacement and deformation of training and testing patterns. Various types of experiments focused on hand-written letter-characters recognition were conducted. The main problem of hand-written letter-characters recognition is the variability of the shapes of the patterns. All types of mentioned deviations such as rotation, displacement, deformation and noise are possible in letter-characters patterns. The Bezier's curves types recognition problem

has the same attributes. It can be considered that there is a high similarity between the hand-written letter-characters recognition problem and the proposed problem of Bezier's curves types recognition. Table 1. shows the best result of experiments focused on formation types recognition using neocognitron network on data consisting of Bezier's curves constructed from randomly selected positions and direction vectors.

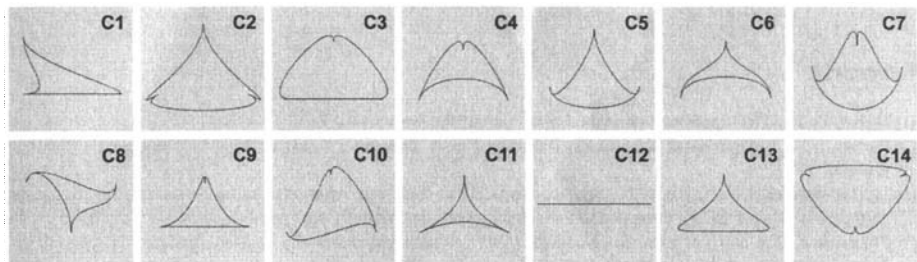


Figure 6. Typical formations of the three robots. 14 typical relevant formation types were identified empirically and used for supervised learning of neocognitron network.

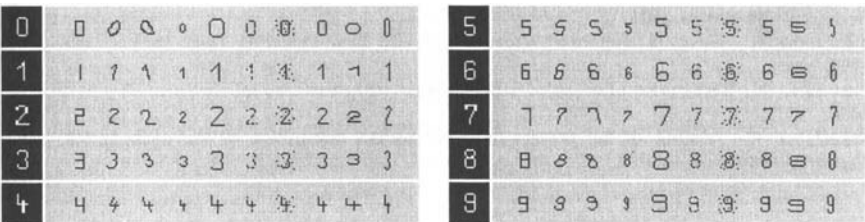


Figure 7. Example of used training/testing hand-written letter-characters. In the dark square on the left side is a typical pattern. High level of noise and shape deviations can be seen.

pattern	c1	c2	c3	c4	c5	c6	c7	c8	c9	c10	c11	c12	c13	c14
c1	78.20%	0.00%	0.00%	0.00%	4.70%	7.70%	0.00%	3.40%	0.00%	0.00%	0.00%	0.30%	9.90%	0.00%
c2	0.00%	89.90%	0.00%	0.00%	2.10%	0.50%	0.00%	0.00%	0.00%	2.20%	0.00%	0.00%	3.40%	0.20%
c3	0.00%	0.00%	75.20%	4.20%	0.00%	0.00%	1.80%	0.00%	0.00%	7.70%	0.00%	0.00%	0.00%	7.20%
c4	0.00%	0.00%	8.40%	82.60%	0.00%	0.00%	0.00%	0.00%	0.00%	6.20%	0.00%	0.00%	0.00%	0.00%
c5	0.00%	4.80%	0.00%	0.00%	82.80%	0.00%	1.00%	9.10%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%
c6	8.90%	0.00%	0.00%	0.00%	0.00%	75.50%	0.00%	0.00%	0.00%	0.00%	14.00%	0.00%	0.00%	0.00%
c7	7.40%	0.00%	0.00%	0.00%	3.30%	0.00%	85.80%	0.00%	9.80%	1.20%	0.00%	0.00%	0.00%	0.00%
c8	4.40%	1.20%	0.00%	0.00%	4.50%	3.90%	1.00%	78.80%	0.00%	2.00%	7.00%	0.00%	0.00%	0.00%
c9	0.00%	0.00%	0.00%	0.00%	2.50%	0.00%	0.00%	0.00%	80.00%	6.90%	0.00%	0.00%	2.10%	0.00%
c10	0.00%	2.90%	11.10%	9.30%	0.00%	0.00%	10.80%	0.00%	7.30%	73.80%	0.00%	0.00%	0.00%	0.00%
c11	3.10%	0.00%	0.00%	0.00%	0.00%	12.40%	0.00%	8.90%	0.00%	0.00%	67.80%	0.00%	0.00%	0.00%
c12	0.00%	0.00%	0.20%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	99.70%	0.00%	0.00%
c13	0.00%	1.10%	0.00%	0.00%	0.10%	0.00%	0.00%	0.00%	3.10%	0.00%	11.20%	0.00%	84.80%	0.00%
c14	0.00%	0.30%	7.10%	3.90%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	0.00%	92.60%

Table 1. The best result of experiments using neocognitron network for formation types identification.

4. Conclusion and future work

The paper presents the implementation and application of biologically inspired neocognitron neural network for the identification soccer robots formation types real-time. The similarity of this task with the hand-written letter-characters recognition problem is presented in the project. The main feature of this kind of problem is high level of pattern shape deviation such as noise, rotation, displacement and deformation. The advantage of used neocognitron network is its robustness against the mentioned types of deviation.

Results of preliminary experiments are presented. Accuracy of results can be improved by a more versatile training sample covering more aspects of the typical shapes but this approach seems to be uncomfortable for practical realization since a lot of instances for each typical formation are required. Further attention will be focused on the representation of high-level features for improved generalization. The further research will be focused also on the utilisation of fuzzy sets in final results interpretation.

References

- [1] Hubel, D. H.: The visual cortex of the brain. Scientific American, 1963.
- [2] Parker, J. R.: Algorithms for Image Processing and Computer Vision. Wiley Computer Publishing, Kanada, 1997.
- [3] Jay G. Rueckl, J.G., Cave, K.R., and Kosslyn, S.M.: Why are 'what' and 'where' processed by separate cortical visual system ? A computational investigation. *Journal of Cognitive Neuroscience*, 1988.
- [4] Fukushima, K., and Miyake, S.: Neocognitron : A new algorithm for pattern recognition tolerant of deformations and shifts in patterns. *Pattern Recognition*, 1982.
- [5] Fukushima, K., Wake, N.: Handwritten alphanumeric character recognition by the neocognitron. *IEEE Transactions on Neural Networks*, 1991.
- [6] Fukushima, K.: Neocognitron : A Self-organizing Neural Network Model for a Mechanism of Pattern. *Biological Cybernetics*, 1988.
- [7] Fukushima, K., Wake, N.: Improved Neocognitron with Bend-Detecting Cells. *IJCNN-92-Baltimore*, 1992.
- [8] Fukushima, K.: Recognition of occluded patterns : a neural network model. Technical Report of IEICE, 2000.
- [9] Fukushima, K.: Analysis of the process of visual pattern recognition by the neocognitron. *Neural Networks*, 1989.

The experimental study of the competitive co-evolution using Predator-Prey tasks

Gyongyike Gebeová¹, Miroslav Hudec³, Peter Kostelník², Vratislav Kováč³

*SWH a Siemens company*¹,

Lomená 1, Košice, Slovakia

gebeova.gyongyike@swh.sk

Institute of Informatics, Slovak Academy of Sciences²,

Dúbravská cesta 9, Bratislava, Slovakia

kostelni@neuron.tuke.sk

Center for Intelligent Technologies³,

Department of Cybernetics and Artificial Intelligence,

Institute of Computer Technology, Technical University of Košice,

Boženy Nemcovej 3, 041 20 Košice, Slovak Republic.

{hudecm,kovacv}@neuron.tuke.sk

Abstract. The aim of this paper is to present the theoretical and experimental study about competitive co-evolution using Predator-Prey task. The objective was to study the dynamics of this approach and also the effect of ontogenetic adaptive change from the perspective of co-evolution of two linked systems - Predator and Prey. This project was tested on simulation of 2 mobile robots test-bed, but application potential is much wider including applications in various parts of finance and economy, self-adaptive control in large scale systems, medicine and evolutionary systems in general. The results confirm some studies which have been made in this domain and point out the Red-Queen effect in the realized experiments.

keywords: *competitive co-evolution, evolutionary robotics, neural networks, predator-prey domain*

1 Introduction

Co-evolution is the process that is observed in relation of two (or more) populations, which are both in evolutionary process, and they influence each other in some way. Individuals of one population interact fitness of other population individuals. Special type of co-evolution, so called **competitive co-evolution** is a type of co-evolution, wherein individuals of a particular population compete for the living space, delimited sources, or they even use individuals from other species for their own benefits and thereby they decrease probability of their survival. The relation **predator-prey** is typical example of this approach. The success of predators implies the failure of prey and vice versa. Evolution of new behaviour in one species represents a new challenge for the other species which is required to evolve new strategies. The fact, that the reproductive value (fitness) of certain trait combination can be affected by adaptive changes in the competing species resulting in a continuous modification of the evolutionary surface, is so-called **Red Queen Effect** (established by L.V. Valen in [10]), a typical phenomenon in a co-evolution process, (see figure 1).

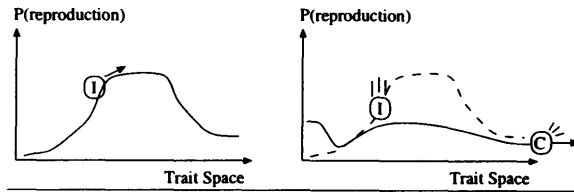


Figure 1: Left: probability of reproduction succes of a single population *I* in static evolutionary enviroment, evolution emerge to higher reproduction success, Right: probability of reproduction in co-evolution proces, fitness of existent combination of traits can be influenced by adaptive change of concurrent population *C*.

This type of relation can be observed in various application domains and could be a matter of theoretical and experimental simulation e.g. in finance [9]. There have been a number of studies in this area [2], [5], [3] [6], [7], [8]. The evolutionary robotics is an interesting test bed to study co-evolution e.g. using predator-prey tasks. Evolutionary robotics is a sub-domain, which has been presented firstly in [1]. The main goal is to design the controller that will give the robot an intelligent behaviour with high degree of autonomy by gathering knowledge about the enviroment and fulfillment of task given to the robot that can be very general e.g. to avoid obstacles, to survive in the enviroment or to hunt a prey. The problem is that the system (robot) is a dynamic system in a dynamic enviroment and these are non-trivial tasks. The control module of a robot is encoded in chromosome and the goal is to determine (find) the most efficient control approach in the wide knowledge space using genetic-like approach.

2 Experimental simulation

2.1 Introduction

Simulation enviroment developed in C++ and graphical visualization was used to follow the actual situation between a predator and a prey. The designed tool gives various possibilities to simulate various aspects of co-evolution phenomena using predator-prey tasks. The experimental setup was compatible with number of similar projects e.g. [2], [3].

2.2 The mobile robots simulator

The mobile robots were represented as complex systems consisting of three basic modules : preception module, neurocontroler and effectors. Both mobile robots was equiped with eight active infrared proximity sensors, six on the front side and two on the back. These sensors could detect a wall at a distance of approximately 3 cm and another robot at a distance approximately 1 cm. In addition, the predator has a vision module, which allows him to sense objects from visual field approximately 36°. This visual field was divided to five 7° sectors, each of them was associated to one photoreceptor. Both robots were placed into 47x47 cm arena with high white walls so that the predator could always see the prey (if within the visual angle) as a black spot on a white background. The control module consisted of a simple neural network without hidden layers with lateral conection on two output neurons. Inputs of NN was associated with sensor inputs of agents, namely 8 inputs (8 infrared sensors) for the prey and 13 inputs (8 infrared proximity sensors and 5 virtual fotoreceptors) for the predator. The output layer consisted of two neurons with sigmoidal activation function. Their outputs defined the speed of left and right wheel of the robot that represented an effector module of the robot. Prey maximum speed was twice that of the predator. The asociation between inputs/outputs of NN and modules of agent and architecture of NN are shown in figure 2.

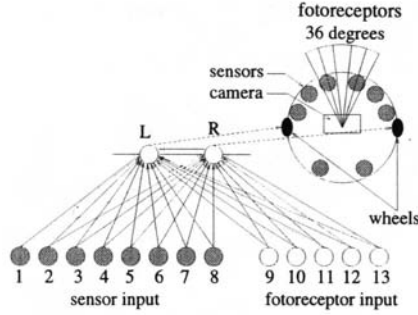


Figure 2: The architecture of a neuro-controller (NC) and association between input/output NC and physical modules of the agent

2.3 Evolutionary adaptation of the mobile robot neuro-controller

The control module of robots was evolved by genetic algorithm. Direct coding of synapses was used, where each of them was coded by 5 bits, one bit for a sign and four bits for value of synaptic weight. Preys chromosome consisted of 100 binary bits and predators chromosome consisted of 150 binary bits. The fitness of an individual was measured through so-called "tournaments" - competitions, where two individuals, the predator and the prey were placed to arena for the time of 500 internal clock cycles. Each individual was confronted with 10 individuals of concurrent species, and the fitness value was then normalised upon the number of tournaments. Two types of fitness function were used :

- **Time-to-Contact fitness function** - individuals were evaluated upon number of internal cycles until the predator caught the prey. Prey fitness ψ_{py} and predator fitness ψ_{pr} can be calculated :

$$\psi_{py} = \frac{1}{K} \sum_{k=1}^K \frac{x_k}{500}$$

$$\psi_{pr} = \frac{1}{K} \sum_{k=1}^K \left(1 - \frac{x_k}{500}\right)$$

- **Distance fitness function** - individuals were evaluated upon the euclidean distance between the prey and the predator after 500 internal cycles. Prey fitness ψ_{py} and predator fitness ψ_{pr} can be calculated :

$$\psi_{py} = \frac{1}{K} \sum_{k=1}^K \frac{d_k}{D}$$

$$\psi_{pr} = \frac{1}{K} \sum_{k=1}^K \left(1 - \frac{d_k}{D}\right)$$

Where K is a number of tournaments, D is maximum euclidean distance between the predator and the prey in arena, $x_k \in \{1, 500\}$ is the number of internal cycles until the predator caught the prey in k tournament, and d_k is the distance between the predator and the prey after 500 internal cycles in k tournament ($k \in \{1, K\}$).

Tournaments served to destine fitness of an **individual tournament** (evaluated individual is confronted with best individuals from several previous generations), or to obtain complex look at dynamics of evolutionary process in **amaster tournament** (best individuals are confronted with best

individuals from all generation after ending the evolutionary process). Evolution of both populations ran during 100 generations.

Entire evolution process can be summarised in to the following steps :

1. Random inicialization of both predators and preys population.
2. Organization of individual tournaments between individuals from predator and prey population (in experiments we use two types of IS, for details see below).
3. Evaluating individuals by fitness values according to TimetoContact or Distance fitness functions.
4. Selection of parents, specifically 20 % of the best individuals from population.
5. Reproduction, offsprings are generated by genetic operators :
 - one point crossover
 - random mutation, i.e. bit switching.
6. Creation of new population from parents and offsprings.
7. Organization of master tournaments for the best individuals from both populations.
8. Repetition of steps 2.-7. 100 times

2.4 Experiments realization

Experiments were devided to two basic groups:

1. Basic experiments

- comparison of influence of two different fitness functions on the dynamics of evolutionary process. "Time to contact" and "Distance" fitness functions were used.
- investigation of influence of different individual tournament (IT) types on the dynamics of evolutionary process. In the first case, each evaluated individual was confrontated with 10 randomly selected individuals from other actual population. In the second case, each evaluated individual was confrontated with the best individuals from last 10 concurrent populations.

2. **Equitable experiments** - they were used to investigate the influence of evolutionary process parameters (population size, genetic operators probability) on evolutionary process dynamics.

2.5 Results analysis

Results of experiments aren't exactly measureable in this area, but we can receive interesting information about the influence of competitive co-evolution on dynamics of evolutionary process from IT and MT fitness value development across generations (shown in figure 3). The fitness value of individuals was acquired by IT (figure 3 bottom). As we assumed, a set of oscillations emerge in fitness after an initial short period, by coverage of the Red Queen Effect. We never observed dominance of one population over the other in any of our experiments. However, the fitness for prey always tended to generate higher peaks due to initial position advantage (even in the case of the worst prey and best predator, the latter will always take some time to reach the prey from its starting position).

Master tournaments (shown in the top of figure 3) tell us two things : At which generation we can find the best prey and the best predator, and at which generation we guaranteed to observe the most interesting tournaments. The best individuals are those reporting the highestst fitness when also the competitor reports the highest fitness. Instead, the most entertaining tournaments are those that take place between individuals that report the same fitness level, because these are the situations where both species have the same level of ability to win over the competitor. The results of our experiments can be summed up in the following paragraphs :

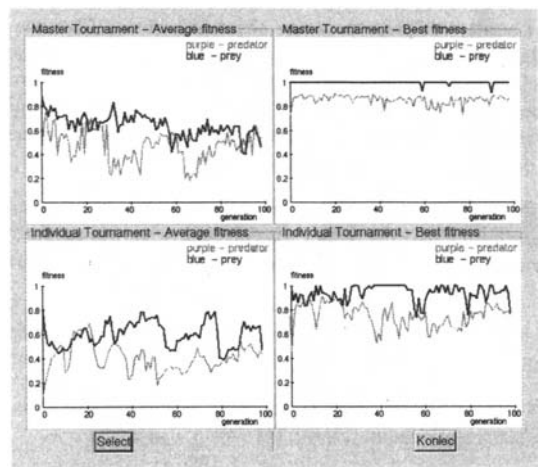


Figure 3: Average and best fitness of predator and prey across generations

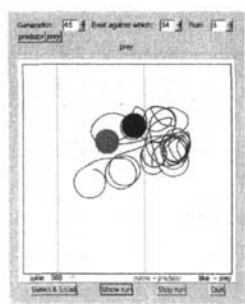


Figure 4: Gen. 45.

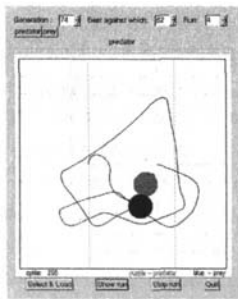


Figure 5: Gen. 74.

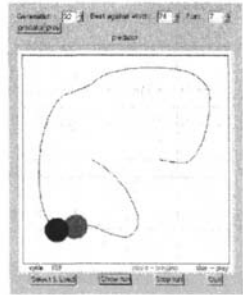


Figure 6: Gen. 92.

- More sensitive strategy of predator can be observed by using **Distance** fitness function instead of **Time-to-contact** fitness function. In figure 4 the prey spins in place and, when the predator gets closer, it avoids it to a small distance. In picture 5 situation, the predator is waiting in place and, whenever it notices the prey, it attacks the prey, is shown. Picture 6 represents predator's so-called "spider strategy". The predator is watching for the prey trajectory and waits for the prey in the place, that is predicted to be visited by prey in future.
- Expressive differences were observable between two types of IT. A minimal difference between robot strategies from each generation was detected, when we confronted individual only with random individual from other actual population. Co-evolution process then converged to trivial solutions.
- Expressive influence of population sizes to the prey and predator ability showed in equitable experiments. Influence of probability of several genetic operators was not achieve.

Typical results of our verification experiments (examples in figure 4,5, 6) are fully comparable with results of experiments mentioned in [2], [6], [7], [8].

3 Conclusion

Overall goal of our experiments was to evolve meaningful behaviour of both the predator and the prey. The predator had to acquire strategies for capturing the prey, and the prey had to acquire suitable strategies for evasion. Agents formed a lot of interesting behaviour like obstacle (static or dynamic) avoidance, visual tracking, object discrimination etc.. In consequence of mobility both of agents environment was dynamic floating. This experiments show, that competitive co-evolution can improve finding strategies for hunting and evasion between the two agents. Competitive co-evolution permits also observation tendency of individuals to recover old facilities after a number of generations with many modifications.

References

- [1] D. Cliff, I. Harvey, P. Husband; Explorations in evolutionary robotics, *Adaptive Behaviour*, 2:73-110. 1993
- [2] D. Floreano, S. Nolfi; God Save the Red Queen! Competition on Co-evolutionary Robotics. *Proceedings of the 2nd International Conference on Genetic Programming*, Morgan Kaufmann, San Mateo, CA, 1977
- [3] D. Floreano, F. Mondana; Hardware Solution for Evolutionary Robotics, *Proceeding of the first European Workshop on Evolutionary Robotics*, Springer Verlag, Berlin, 1998
- [4] G. Gebeová ; Master thesis : Využitie neurogenetických systémov pri riadení autonómnych mobilných robotov, Košice, 2001.
- [5] S. Nolfi, D. Floreano; Learning and evolution. *Autonomous Robots*, 7(1), forthcoming. 1999
- [6] S. Nolfi, D. Floreano; Co-evolving predator and prey robots: Do 'arm races' arise in artificial evolution ?, *Artificial Life*, 1998
- [7] S. Nolfi, D. Floreano; Adaptive behavior in competing co-evolving species, *proceedings of the Fourth European Conference on Artificial Life*, MIT Press, Cambridge, MA, 1997b
- [8] S. Nolfi, D. Floreano, F. Mondana; Co-evolution and Ontogenetic Change in Competing Robots, *Robotics and Autonomous Systems*, to appear. 1998
- [9] Ch. D. Ronin, R. K. Below; New methods for competitive coevolution, technical report #CS96-491.
- [10] L. V. Valen : A New Evolutionary Law, *Evolutionary Theory*, , 1: pp. 1-30, 1973.

Two-dimensional Hologram Model and Computer Simulation of Patterns Processing, Storing and Restoring

Aurelia Profir, Elena Boian, Cleopatra Zelinschi
Institute of Applied Physics, Academy of Sciences of Moldova,
Academiei str. 5, Chişinău, Moldova, MD-2028,
email: aurelia@cc.acad.md

Institute of Mathematics and Computer Science, Academy of Sciences of Moldova,
Academiei str. 5, Chişinău, Moldova, MD-2028,
email: lena@math.mdi

State University of Moldova,
str Mateevici, 66, Chişinău, Moldova, MD-2012,
email: cleozeli@yahoo.com

Abstract. In this work we propose the method of computer simulation of processing, storing and restoring patterns using the hologram principles of information updating.

Keywords: *Neural network, Method of recognition, Storage and restore of patterns, Holographic principles of information processing, Finite automaton, Genetic molecular triggers*

1 Introduction and problem definition

In this work we propose the method of computer simulation of processing, storing and restoring patterns using the hologram principles of information updating. A higher level of parallelism characterizes this method. It is known that there is a specific similarity between the patterns processing in neural networks and an optical hologram [1, 2]. We generalized the existing models [2] based on pseudo-optical hologram principles in which signals of electric nature are received. We propose a model of optical hologram when the signals of wave nature (UV diapason) are propagated in neural networks.

Different forms of self-organization are widely spread and play a major role in the processes of receiving, processing and transmitting information in biosystems [3]. A living cell represents a complex biosystem in which processes of self-regulation and self-organization occur by means of feedbacks and nonlinear properties of system [4, 5]. We try to examine from a new point of view and to adapt this hologram model to the cell level (membrane and DNA processes) for molecular-genetic triggers (MGT) or finite automaton systems. We suppose that the system can receive two types of signals. In the first case, the system receives signals of molecular nature. In the second case, the signals of wave nature of resonant frequencies can be utilized [5].

2 Used methods and approaches

From the synergetic point of view the information system must have same special properties. In order that the system may contain information it is necessary to be multistationary, i.e. to exist in some stationary states, some of them must be stable. This means that the system is dissipative, i.e. processes of self-organizing of structures occur [3]. The information system should be bistable, it should have two stable states (attractors) at least. MGT (self-organizing structure) posses two stable states (attractors) and can switch from one stable state to another. These two stable states of MGT correspond to active and inactive states of genes. MGT can be represented also as a deterministic finite automaton [4, 5]. We generalize the linear model [2] based on hologram principles to create a two-dimensional model for storage and restore of two-dimensional patterns (for example, rhombus) and also for obtaining a better hologram quality. Our model consists of four levels: light source, two-dimensional pattern, two-dimensional hologram, restored two-dimensional pattern. The potentials of active components (MGTs/automatons or neurons) of hologram level are calculated using the differences of phase of received coherent signals from light source and pattern level. In calculation and optimization of main parameters of the model (limited value, input intensity, potentials, distances between the levels and between active components on the level, etc.) we use a suitable algorithm and methods of computer simulation. We implement the investigated model using the possibility to simulate parallel processes in Delphi system in Windows environment.

Conclusion

In this work we propose the method of computer simulation of processing, storing and restoring patterns using the optical hologram principles of information updating. The living cell can be considered as a parallel self-organizing system producing, receiving and transmitting the patterns. Therefore we try to investigate from a new point of view and to adapt this two-dimensional hologram model for neural networks and also at the cell level (membranes and DNA processes).

References

- [1] Heerden P.J.van. The Foundation of Empirical Knowledge. Netherland:N.V. Uitgeverij Wistik-Wassenaar, 1968.
- [2] Kuznetsov O.P., Shipilina L.B. Pseudooptical neural network as a full linear model and its behavior calculation methods. - *Izvestia Akademii Nauk. Teoria i sistemi upravlenia*. Moscow, 2000, Nr.5, pp.168-176. (in Russian)
- [3] Prigogin I. Dissipative Structures and Biological Order. *Adv. Biol. Med. Phys.* 16, 1977, pp. 99-113. 3.
- [4] Kovarskii V.A., Profir A.V. Trigger effect of the λ -phage genome switching. *Mol. Biol.*, 25, 1991, pp. 1293-1300 (in Russian).
- [5] Profir A. The system of molecular-genetic triggers as self-organizing computing system. - *Computer Science Journal of Moldova*, 2001, n. 2, pp.54-71.

IV. Applications

This page intentionally left blank

Evolutionary Human Support Agents and Applications

Toru Yamaguchi, Akira Sannoh*, Mika Iori**

*Department of Electronic System Engineering, Tokyo Metropolitan Institute of Technology
PRESTO, Japan Science and Technology Corporation*

/Department of Electronic System Engineering, Tokyo Metropolitan Institute of
Technology*

6-6, Asahigaoka, Hino, Tokyo, 191-0065 Japan

Abstract. In these years, scientific technology has been evolving, and is used in many fields. But, accompany with deepening of the technology using sophisticated technology needs specific technology. Because of the reason, the technologies that enable us to use the sophisticated technology are required. It is needed that various people can use sophisticated technologies to be used in many fields. The needed technology is Intelligent Agent. The technology makes people who have no skill use sophisticated technology. In this paper, we researched two Intelligent Agents using chaotic retrieval.

Keywords: Neural Network, Chaos, Intellectual Agent

1. Knowledge Generation for Intelligent Agent Using Evolutionary Chaotic Retrieval

1.1 Introduction

Today, information technology is used in many fields. The technology's aim is supporting human. But, using the technology is not easy for the public. For that purpose, Intelligent Agent is needed to be uses in the technology.

In the Agent of the above description, the parts of which works between men and a machine is the most important in man-machine interface. Especially, to display pictures and letters for men is the most important role. A conventional method is one-sided, such a information method is not exactly suitable for a man.

The problem is, this method is appropriate to inform imminence, but is unsuitable to inform the problem which is not pressing. In short, Intelligent Agent must get a picture, and decide how to support a man.

In this section, we generate knowledge for the Intelligent Agent, and aim at making the Intelligent Agent, which can behave in a suitable way.

For one such an example, we researched an Intelligent Agent which works at ITS, similar to Human Support Robot.

1.2 The rough outline of this system

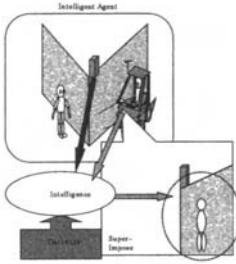


Figure 1. Intelligent Agent System

In this paper, we are studying the output part, which plays a role as an agent of the display for drivers.

This paper proposes an agent system of the display for drivers (Figure 1).

This system consists of three agents:

The agent watches a driver's behavior

The agent does chaotic retrieval

The agent displays selected pictures

This agent system of display for human gets information from the input part, the information obtains some pictures. The agent shows that pictures to a driver, and the driver chooses pictures that are used as the interface between a man and the machine.

Like this agent system, consulting with the user is a good method how to get suitable pictures that are used in the system.

To materialize this system, we used a chaotic retrieval method because of the after-mentioned reasons.

1.3 Chaotic Steepest Descent method

A chaotic Steepest Descent method (CSD method) is the method of chaotic retrieval that creates new knowledge. This method allows chaotic wandering in the minimum energy area by applying a periodically variable non-linear resistance to a scatter of the dynamics formula. It is for the movement at the energy curved surface on the neural network. By using this method, the agent can create new display knowledge. (Figure 2)

Moreover, the CSD method can easily change the characteristic of non-linear resistance.

The CSD method cuts it fine by using next equations (1), (2).

$$m\ddot{u}_i + f(\dot{u}_i, \omega t) = \varepsilon \sum_j w_{ij} a_j + \eta I_i + \begin{cases} 0 & (-\beta \leq u_i \leq \beta) \\ -\alpha(u_i > \beta) \\ \alpha(u_i < -\beta) \end{cases} \quad (1)$$

$$f(\dot{u}_i, \omega t) = [d_0 \sin(\omega t) + d_1] \dot{u}_i + d_2 \dot{u}_i^2 \operatorname{sgn}(\dot{u}_i) \quad (2)$$

In above equations, u_i, a_i, I_i each means i th node's inner condition, input value. Parameter w_{ij} means the weight from i th node to j th node. Function $f(\cdot)$ means non-linear resistance, and m, ε express as nonnegative integer, ω expresses a parameter which is used as a parameter that varies non-linear resistance's trait. Parameter α compensates an energy gradient. Parameter β means the range of absolute value that is node's inner condition and η means a weight coefficient toward input value, d_0, d_1 means the coefficient of linear resistance, d_2 means the coefficient of non-linear resistance.

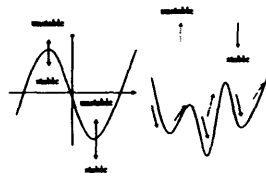


Figure 2. Characteristic variation of non-linear resistance and chaotic wandering on energy curved surface

Function $\text{sgn}(\cdot)$ means a signless function. However, in the equation (1), an input term (ηI_i) is added into the original CSD method.

Discriminating feature of the CSD method is that in the equation, parameter d_2 , which varies the train of non-linear resistance, can varies the degree of chaotic trait. Wherethrough the trait, a user can get new data within the limit that the user sets up.

1.4 Application of Chaotic Retrieval to human interface

In this paper, we propose a driver support agent system, which is implemented in chaotic retrieval. The system indicates suitable pictures for drivers attention to remind.

At the first setout, the system has stored preconcerted data about pictures. This data consist of information on a pixel color and a picture's size. Each pixel color is determined by the RGB value. Each RGB value is presented in 8bit(0~255). And size is presented in 4bit (0~1.875, basic size is 1). Using the preconcerted pictures, the agent creates new data about pictures and in chaotic retrieval. Next, the user chooses one of the created pictures. The agent uses the selected picture data to do chaotic retrieval.

When the driver does chaotic retrieval for the first time, sets the parameter d_2 cubicle to strengthen the non-linear characteristic. And the driver does it for the second time, we make parameter d_2 larger than before. The reason is to weaken non-linear characteristic that makes the agent create a new picture, resembles the picture which was selected by election driver selected. This method enables the agent to create pictures which are suitable to all drivers.

The picture under this dialog (Figure 3) is made in Xwindow Systems. The upper picture approves variations of color, the under picture approves its size.

The agent goes over the process and creates new picture data – a soft touched picture–, which is suitable for drivers.

Next thing to do is to construct an agent system by using this algorithm. In this system, we used 3-D stenography, which is made by OpenGL, as a simulation to show some situations. At the same time, we imposed pictures, which were obtained by chaotic retrieval.

Figure 4 is the system's rough outline.

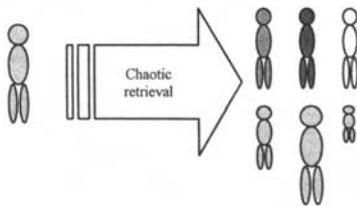


Figure 3. Created pictures

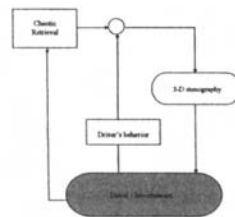


Figure 4. The proposed system's rough outline

When this system is used, it indicates pictures connected with the driver's behavior. An illuminate indication is needed when the driver is not aware of pedestrians. In this system, the driver enables to sniff danger by using the method like the under picture.

First, a CCD camera copies driver's face squarely. The system searches for the driver's face and a lip position. Next, from the physical relationship of the face and the lip, the system judges whether the driver is aware of pedestrians or not. If his lip was in a biased position right or left, it means he wasn't watching the opposite direction. (Figure 5)



Figure 5. A driver's face and his attention

In this case, a pedestrian comes from the direction where the driver does not have an eye to, so the agent must remind the driver of it. Conversely, the information that is known to the driver is not needed to impart the driver. Due to this method, the information displayed to the driver could be minimum.

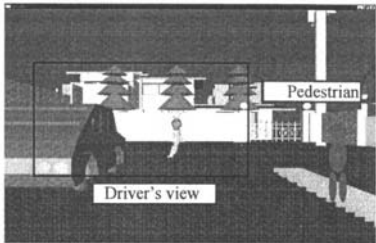


Figure 6. A Super-imposed picture

Under this indication method, the agent shares significant information to prevent the driver from being confused by too many pictures and also from overlooking important information. The picture under this sentence is the simulation image. (Figure 6.) In this picture, the driver's attention is fixed on the Drivers view. The agent judges to notify another pedestrian for the driver, who is walking toward the car.

1.5 Evaluation and Consideration

Four experimental subjects evaluated this agent system. The evaluated point is like that.

1) The picture is legible

2) The picture is soft-touched

Each picture is evaluated on a five-point scale.

These evaluated pictures are of three types. One is created by the agent system and selected by an experimental subject, another is prepared by the system builder, and another is accentuated, which is generally used to attract human's attention.

The evaluations are shown in tables 1,2.

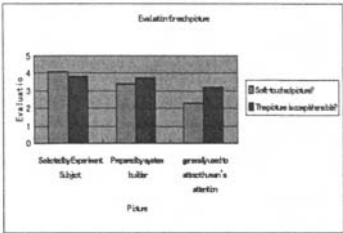


Table 1. The picture is soft-touched one or not

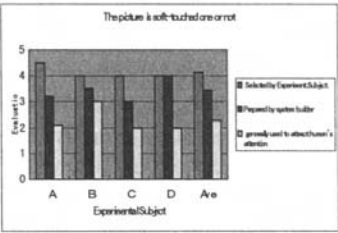


Table 2. The picture is comprehensible one or not

The pictures that were created by chaotic retrieval received recognition that the pictures were soft-touched pictures. On the other hand, the pictures were not rated that the pictures are comprehensibly high. But two contrary attributes were achieved by using this method. (Table 3)

Therefore, the pictures that were created by chaotic retrieval are not bright-colored and comprehensible in this simulation system.

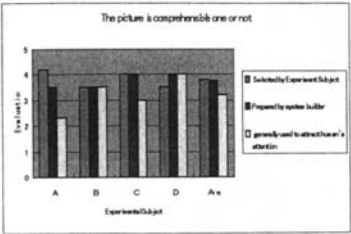


Table 3. Evaluation for each picture

2. The chaos evolution dialog agent adapting to 3D multiplatform

2.1 Introduction

Next, we propose the 3D graphics display system which does not depend on a screen product, the resolution degree, and the drawing process by Chaotic Retrieval.

2.2 The rough outline of this system

As for this system, it is composed by two computers (Figure 7), and on the other hand, the program of OpenGL (which CSD is the Chaotic Retrieval part) and 3D display part is installed on Windows.

The other is connected to the camera, and detects the inclination of user's face on Linux. To pass the display part information, Ispace is installed.

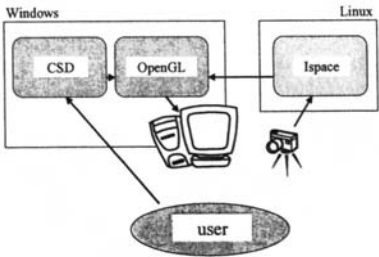


Figure 7. Composition of the system

2.3 Parameter changed by Chaotic Retrieval

To change the display pattern, the element which the feature of the image is decided about is changed by Chaotic Retrieval as a parameter. To the parameter RGB, each values the

color for clothes of the agent (monkey), and the sizes of eyes, the ear, and the body. In addition, the flag whether to put the texture on the object of the background, the field of view, the number of division when approximated to the polyhedron the sphere, and the distance in drawing the furthest points.

The data passed to CSD when Chaotic Retrieval is done when a bit of 26 digits. It is concerning the agent by 14 digits in that. 12 digits of the remainder are concerning the background (or, the entire image).

Parameters other than the texture change into four stages to 00-11 by allocating all 2 bits. As for the parameter of the texture, the 1 bit corresponds to each object of the background (floor, ceiling, stairs, and wall, etc.). When the bit indicates 1 the texture exists, when 0, there is none.

Like this by the change of the parameter; we can obtain 3D animation of the quality which corresponds to the spec of the computer.

2.4 Application of Chaotic Retrieval to actual display

The flow by which the display pattern of the 3D graphics is omitted because it is the same as the above-mentioned driver support agent system.

2.5 Evaluation and Consideration

I experimented with SGI which connected the plasma display and a mobile computer. The experimental subject evaluated the pattern which had been obtained by the initial pattern (another one was prepared with SGI and a mobile personal computer) prepared for the system and Chaotic Retrieval by five stages. (Table 4)

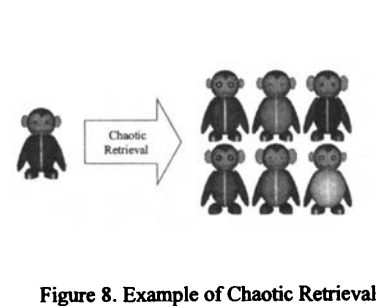


Figure 8. Example of Chaotic Retrieval

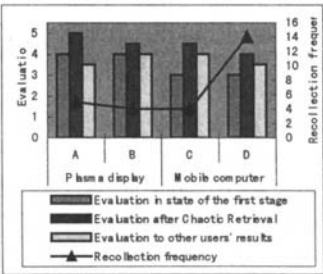


Table 4. Evaluation for each picture

All experimental subjects were given a high evaluation from the initial pattern to the pattern by Chaotic Retrieval. The recollection frequency has been installed within 6.75 times on the average and the practicable ranges, too.

When the experimental subject's impression is heard, the attention is paid to the change in agent's face and the change in the background etc. is not noticed easily while experimenting.

3. Conclusions

The 3D animation in the mobile computer is not widespread now, and it is thought that such a demand is generated in the future enough.

Because the difference in the performance surface of the computer is not lost, I think that this research was able to show the one solution method.

*This research is supported by **PRESTO**.*

References

- [1] M.Takahide, N.Kohata, T. Yamaguchi : Intelligent Space and its Application to Man-Machine System, Proc.of IEEE Internationalconference on Systems, Man and Cybernetics (SMC'99), Tokyo, Japan,(1999,10)
- [2] T. Yamaguchi, H.Nitta, K.Hirayama , N.Kohata, T,Takagi :Human Centered Network Architecture for ITS, Proc. Of The Fourth Asian Fuzzy Systems symposium(AFSS2000) vol.1, pp.468-471(2000)
- [3] Naoki Kohata, Toru Yamaguchi, Takanobu Baba, Hideki Hashimoto : Chaotic Evolutionary Parallel Computation on Intelligent Agents, Journal of Robotics and Mechatronics, Vol.10, No.5, pp424-430(1998)
- [4] T. Yamaguchi, K.Hirayama, M.Takahide, H. Hashimoto : Intelligent Space and its Application to Augmented Sensing, Proc. of IEEE/IEEJ/JSAI Conference on Intelligent Transportation Systems(ITSC99),Tokyo,Japan pp901-906(1999.10)
- [5] Kaoru Nakano :foundation of Neuro Computer, Tokyo, CORONA PUBLISHING CO., LTD, 1991
- [6] Shun-ichi Amari:Neural Network Model and Connectionism, Tokyo, UNIVERSITY OF TOKYO PRESS,1989
- [7] J.J.Hopfield&D.W.Tank:Neural Computation of decisions in optimization problems. Biological Cybernetics,52,141(1985)
- [8] E.H.Mamdani:Applications of Fuzzy Algorithms for Control of Simple Dynamics Plant, Proceedings of IEE, Vol.121,No12, 1585-1588(1974)

Distributed Learning in Behaviour Based Mobile Robot Control

Peter Kostelník^{1,2}, Miroslav Hudec¹ and Marek Šamulka^{1,2}

Center for Intelligent Technologies

Department of Cybernetics and Artificial Intelligence¹,

University of Technology Košice, Letná 9, 040 01 Košice, Slovak Republic

Institute of Informatics Slovak Academy of Sciences²,

Dúbravská cesta 9, 842 37 Bratislava, Slovak Republic

{kostelni,hudecm,samulka}@neuron.tuke.sk

Abstract.

In this paper, a behaviour based single agent system is described which is able to act and learn in an unknown, dynamic and noisy environment. The basic idea of the proposed architecture is that the sensory information from the real world is clustered, where each cluster represents a situation in the agent's environment, then to each cluster or group of clusters an action is assigned via reinforcement learning. In this work for clustering the Kohonen's SOMs are implemented and in the state-action pair – behaviour creation the Q-learning is applied. The proposed architecture will be applied in the robotic domain, to the task of navigation in an unknown environment with learning of a simple avoidance behaviour. For this purpose a simulation/real environment was created and is also briefly discussed. Results of experiments show that our system is able to learn the desired behaviour in a few trials.

keywords: *behaviour based control, neural networks, reinforcement learning, unknown environment, robot simulator, distributed learning*

1 Introduction

During the past years there has been an increasing demand on systems that can automatically act in and adapt to an unknown environments. Very popular and dynamic research area is the robotics domain, which has the properties of an unknown, dynamic stochastic environment. Most of successful navigation, collision avoidance algorithms usually require availability of maps and avoidance rules, usually manually generated. In behaviour based systems—robots, the robot controller consists of behaviours each of which maintains a specific goal (the goal of "avoid-obstacles" behaviour is to prevent robot from collision with other object). A behaviour based robot is controlled by a structure of interacting behaviours. Such a structure lets us provide new (behaviour) modules to the robot and organize them hierarchically. The behaviours are usually initiated/activated as a response to internal or external conditions [4].

Behaviours are described in terms of reinforcements provided by the environment for realizing an action in the current state. The reinforcement signal defines the desired behaviour in terms of reward and/or punishment i.e. what state of the world is acceptable and which is wrong for agent or robot [1]. Reinforcement signal, together with the sensory information, is the only feedback from the environment. Use of reinforcement feedback lets us focus on learning system which will be independent from the environment's specifications.

1.1 Motivation

Our previous work [2] was focused on comparison study of different intelligent control methods e.g. fuzzy system [3] and neural system on task of parking in the known environment. That approach was based on prescribed/predefined navigation rules for robot via fuzzy rules, or via predefined set of paths for supervised learning (the obstacle avoidance was performed via external algorithm). The neurocontroller had shorter and smoother paths compared with fuzzy controller. However, in real world problems, usually, we do not have full description of the world and the environment provides only rough nondescriptive feedback.

We also performed experiments with reinforcement learning with satisfactory results: the optimization of the path was successful and in addition the reinforcement system was able to find the route to goal from any, previously not defined-considered, place. However, the initial phase took too much time - the robot had to learn lower level knowledge about the environment (to avoid an obstacle) as well as higher level knowledge (optimize trajectory to the goal) in the same system. In the present work we divided the goals into the hierarchy of modules-behaviours as it is known in behaviour based systems i.e. subsumption architecture.

2 The Overall Control Architecture

When robot has to act in a dynamic environment or in an environment where some changes are expected, then he has to have mechanism which will adapt the robot to those changes. In such environment new situations may appear and thus the robot requires a mechanism that systematically processes and incorporates new information about the environment. We suppose a goal/task oriented operation/acting of agents, where each agent should be responsible for solving one or more tasks at the same time. In this case the use of behaviour based architecture should be advantageous, because it allows application of multiple independent goal oriented modules simultaneously. The benefit of the behaviour based system is the ability of internal state storing and therefore the ability of learning while each behaviour module can be adapted independently, working with private world representation. Accordingly, the knowledge is distributed in the whole control system by mutually interacting models of the world.

When a dynamic change of the environment occurs, the use of independent knowledge allows local adaptation of modules, responsible for defined goal/task, in a distributed manner.

The proposed architecture consists of three parts:

- The perception component processes sensory inputs and creates concepts of typical situations rising from robot-world interactions. The Kohonen neural network creates cluster structure from sensor activation values. The set of clusters represents world's semantic. Each neuron in Kohonen neural network is the center of the cluster and is active for set of activation combinations on sensors. We assume that unsupervised learning enables the robot to continually adapt the perception concepts when new situations in the environment appear.
- The behavioural component processes the perception information and learns the robot control policy using reinforcement feedback. It may consist of many independent modules, each representing a goal/task of a behaviour. Each behaviour module is responsible for achieving (specific) goal defined by its reinforcement function. Task/goal achieving process is realized by learning the action selection mechanism with Q-learning algorithm. Behavioural component can be viewed as a set of independent goal/task achieving modules learning independently in a distributed manner. Reinforcement Q-learning algorithm enables the robot to adapt the action selection strategy in a changing environment.

- The top-level control component is realized as a subsumption network. It takes outputs from modules of the behaviour layer and arbitrates the activation of single independent behaviours according to their priority explicitly defined in subsumption relations. Generally, the subsumption network is responsible for behaviour selection. Whenever more behaviours become active, the subsumption network selects behaviour with the highest priority. The action, generated by selected behaviour, is applied to the robot actuators.

The perception and behavioural components are implemented as connected artificial neural networks. The subsumption network is implemented as a simple system of production rules. The overall control architecture design is shown on Fig. 1.

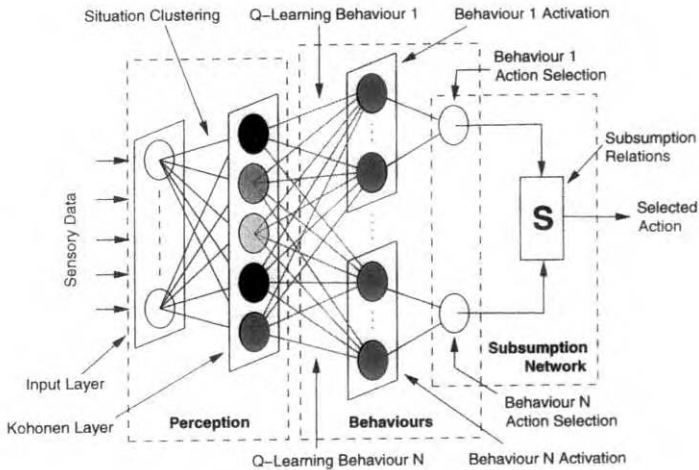


Figure 1: Overall control architecture design.

3 The Robot Simulation

To simulate various groups of mobile robots achieving various collective tasks we created a simulator that provides modelling of the environment, objects, robots and inter-robot communication.

Each simulated robot is controlled by one agent, connected to simulator from remote host by TCP/IP, in client/server fashion. Thus robots can be controlled by agents implemented in various programming languages and running under various operating systems.

The purpose of creating new simulator is twofold: (1) we needed an extensible and fast simulation environment and (2) we wanted to have the same server for running simulations of the robots as well as for communication with real robots and so hide the real dynamics from the agents.

Therefore the multi-robot simulation system can run in two different modes:

- Simulation mode: System running in this mode works as the standard simulator modelling environment, objects and robots.
- Real-world mode: This mode enables to control the real LEGO robots in the real world. Next extension of the system is the application of camera sensing the environment, extracting data and updating the world model.

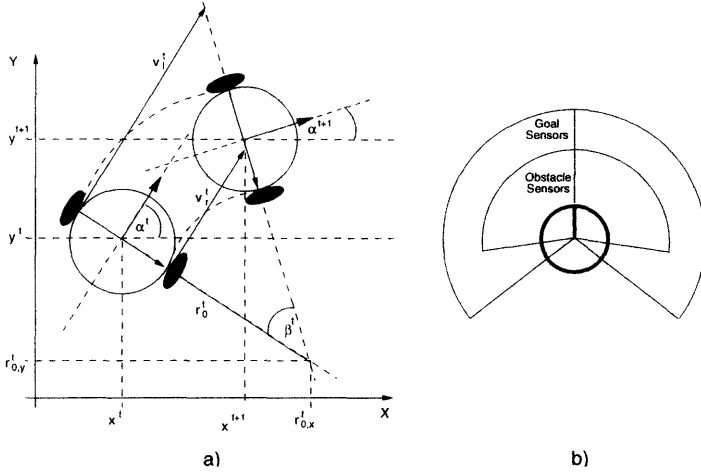


Figure 2: a) Model of dynamics for two wheeled robots. b) Simple model of used sensing system with two obstacle and two goal sensors on each side of the robot.

Visualization of system performance is done using a special type of client - the Monitor - connected to the action server.

The internal representation of the robot models is a part of the server world representation structures. Each model of the robot consists of three main parts:

- **Perception model:** It includes representations of used sensors and the sensing parameters. Proposed robot architecture uses the simple sensor model with two sensing types: (1) the obstacle sensor and (2) the goal sensor. Each sensor returns the inverted value of its activation (i.e. distance from the object). The used model of sensing system is shown in Figure 2b.
- **Model of actuators:** It consists of robots construction types. The proposed robot architecture implements the model of two wheeled robots with two actuators: the left and the right wheel. The model of the robot's dynamics is shown in Figure 2a.
- **Robot parameters:** It includes the main architecture properties of the robots such as scale of the robot, sensing radius, etc.

Each agent connected to the action server controls one robot by sending control messages depending on the used robot actuator model. The simulator system propagates control codes to robots and computes their new positions and directions in the world model. After application of the control codes simulator returns local world information for sensing fields of the robots. Perception module computes activations of the used sensors and constructs the sensor vector.

New position and direction coordinates x^{t+1} , y^{t+1} and α^{t+1} of the robot can be computed from the actual position vector and speed signals $[v_l^t, v_r^t]$ for the left and right wheel using following equations:

$$\begin{aligned} x^{t+1} &= x^t + (x^t - r_{0,x}^t) * \cos(\beta^t) - (y^t - r_{0,y}^t) * \sin(\beta^t) \\ y^{t+1} &= y^t + (y^t - r_{0,y}^t) * \cos(\beta^t) - (x^t - r_{0,x}^t) * \sin(\beta^t) , \\ \alpha^{t+1} &= \alpha^t + \beta^t \end{aligned}$$

where, x^t , y^t and α^t are the actual position and direction of the robot. $r_{0,x}^t$ and $r_{0,y}^t$ are the actual position coordinates of robot rotation center. β^t is the actual angle of the robot rotation.

If one needs a highly accurate or special model of robot dynamics this simple model could be easily replaced with more sophisticated one.

4 Implementation

4.1 Perception Component

The perception system of the proposed architecture is realized as the Kohonen neural network clustering the activation values of the input sensor vector. Set of neurons in the perception system (the centers of the clusters) represents world semantics as the set of typical situations arising from robot-world interactions.

The model of the robot uses two pairs of sensors: (1) the goal sensor pair and (2) the obstacle sensor pair. Each pair of sensors uses different sensing range and the activation of each sensor is an analogue value. Thus the space of possible world states is an infinite set of possible sensed combinations. Using continuous propagating the sensed values through Kohonen network causes permanent moving of the cluster structure centers. One of the possible solutions to this problem is using the set of threshold values for sensor activations. Thresholding of the sensor activation values causes reduction of possible world states and it restricts movement of the cluster structure centers. The analogue activation value of the sensor is transformed to the set of possible threshold values.

Weights of Kohonen's network are updated according to:

$$\begin{aligned}\Delta w_{ij} &= \eta \Lambda(i, i^*) (x_i - w_{ij}) \\ \Lambda(i, i^*) &= \exp \left(-\frac{d_{i,i^*}^2}{2\sigma^2} \right)\end{aligned}$$

where $\Lambda(i, i^*)$ is the neighbourhood function. The neighbourhood function monotonously decreased with time.

4.2 Behaviour Component

Behaviour component provides mapping between typical states of the world and actions in the corresponding module. The mapping is done using Q-learning [1] update rule:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right]. \quad (1)$$

An agent acting in the real world should be able to adapt without any kind of global knowledge about the world. Therefore the function generating the reinforcement signals is derived only from the change of sensor values representing the implicit knowledge about new world state. Reinforcements for both of implemented goals/tasks are defined as positive or negative change of activations on sensors. The reinforcement functions for *obstacle avoidance* behaviour (Eq. 3) and *simple parking* behaviour (Eq.4) are defined as follows:

$$Success(t) = \sum_{i=0}^N \Delta Sensor_i(t) \quad (2)$$

$$r_{avoid}(t) = \begin{cases} -1 & \text{if } Success > 0 \\ 1 & \text{if } Success < 0 \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

$$r_{find}(t) = \begin{cases} 1 & \text{if } Success > 0 \\ -1 & \text{if } Success < 0, \\ 0 & \text{otherwise} \end{cases} \quad (4)$$

where $r_{avoid}(t)$ is the reinforcement signal defined for *avoid* behaviour and $r_{find}(t)$ is for *find* behaviour. The *Success* defined in Eq.2 represents variation of the sensor activations caused by change of the world after applying the last action. Both of the reinforcement signals can be interpreted in the terms of positive or negative changes on sensors: (1) for *obstacle avoidance* behaviour, if the robot has higher activation values on the sensors after action application i.e. it is approaching the obstacle and receives the punishment; otherwise it receives a reward; (2) for *find* (or *park into*) behaviour the situation is inverted, if the activation values on sensors after applying the action are higher, the robot is approaching the goal and receives a reward; otherwise it receives a punishment. If there is no change on the sensors, the robot receives *reinforcement* = 0.

The Q-values are stored in connections between neurons in Kohonen layer (clustered situations) and neurons in behaviour representation (each neuron representing one action). In the behaviour component there are two behaviours, each behaviour consists of five neurons representing five actions applicable on actuators: *Forward*, *Turn left*, *Rotate left*, *Turn right*, *Rotate right*. All actions are defined as the signals for left and right wheel motors.

4.3 The subsumption network

The toplevel control component decides which behaviour will be, in the current situation applied. This component is implemented as subsumption network, with predefined priority for each module. This priority as well as the definition of the reinforcement signals is given by robot constructor and it defines the purpose of the robot. The used subsumption network is shown on Fig. 3. The robot control

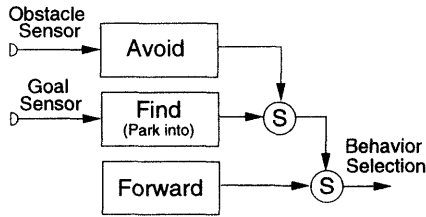


Figure 3: The subsumption network for simple parking problem.

consists of three defined behaviours organized by their priority. The highest priority is set for *Avoid* behaviour that is active whenever the obstacle is sensed. If there is no need to avoid obstacle and the attractor point is sensed, the *Find* behaviour is active and the robot approaches the goal. When there is no activation on sensors, the *Forward* behaviour is active and the robot begins to track the environment. The *Avoid* and *Find* behaviours are implemented as neural networks with Q-learning, *Forward* behaviour is a simple action directly applied to actuators. Whenever more behaviours in the subsumption network are active the one with the highest priority is selected and action generated by selected behaviour is applied to robot actuators.

4.4 The Control Loop

The agent works in a loop and one iteration can be described as follows:

- Action selection stage:
 1. Input layer of Kohonen network is provided with actual values of sensors.
 2. The winner is selected and the weights are updated.
 3. Simple forward propagation from winning neuron to behaviour layer follows.
 4. Each behaviour is passed through the subsumption network. The behaviour with the highest priority is selected.
 5. ϵ -greedy action selection [1] on the selected/winning behaviour is performed and the selected action is performed on actuators. The ϵ -greedy selection is implemented as roulette.
- State-action mapping stage:
 1. Reinforcement signal $r(t)$ is observed by Eq. 3 or Eq. 4 for the applied behaviour.
 2. Changes to the weights representing the applied behaviour and action are computed by Eq. 1 and the weights are updated

This loop can be finite if the task is episodic (find an attractor) or infinite if our task is continuous (space surveillance). The setting of parameters for Kohonen network and reinforcement learning is up to robot designer.

5 Experiments and Results

Provided experiments can be divided to three parts. First two experiments investigated the operation of two algorithms separately (i.e. clustering ability and Q-learning performance) and the third set of experiments investigated the full system's operation. The experiments can be described as follows:

5.1 Situation clustering

This set of experiments was focused on cluster stabilization testing by applying various parameters of learning and different network sizes. As the experiments were focused on situation clustering, the robots were controlled by simple predefined reactive architecture realizing safe wandering in the environment in order to gather the learning patterns. The objective of this set of experiments was to observe the emergence of the world semantic arising from robot-environment interactions. We assume that there is no explicit objective measure for success evaluation of the final clusters configuration. One of possible ways of the results interpretation is the observed rate of the space coverage. Implemented robot architecture consists of two sensor pairs with different sensing types: (1) the obstacle sensor and (2) the goal/attractor sensor (see Figure 2b for more details). Thus clustering should be done in a four dimensional space. Because of the similarity of sensors it is possible to decrease dimension of the input space to 2 dimensions, without losing of the representation ability. The input pattern construction for situation clustering was realized in dependence of actually active behaviour selected by subsumption network: (1) in the case of obstacle avoidance behaviour the pair of goal sensors was used; (2) in the case of simple parking behaviour the goal sensors were used. The size of Kohonen layer was experimentally determined and fixed to ten neurons. We performed two kinds of input pattern construction: (1) analogous sensor values normalized to interval $< 0, 10 >$; (2) thresholded sensor values where each value was transformed to the one of the 6 possible values $\{0, 2, 4, 6, 8, 10\}$. Input patterns were scanned in each m -th iteration where m was randomly set to $m \in \{3, 5, 7\}$. The cluster

structure was not able to stabilize even after 2000 learning steps. This problem can be explained in the terms of environment dynamics. It is not possible to ensure that the full set of situations in the world will be explored by the robot. This fact causes a high probability of possible future cluster structure change.

Figure 4 shows a typical example of emerging cluster structure for world semantic representation. Example Figure 4b shows usual space coverage. The centers of the clusters are well distributed.

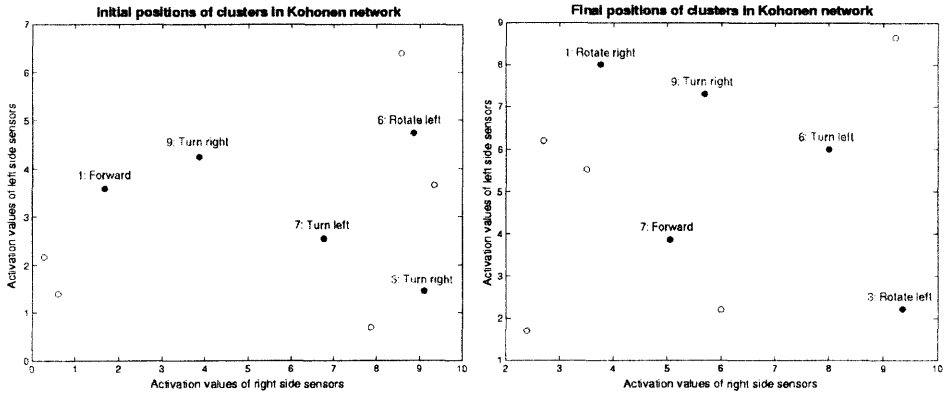


Figure 4: Stabilization of cluster structure in perception layer with examples of actions assigned to situation clusters.

5.2 Distributed learning of behaviours

In this set of experiments fixed sensoric concepts represented by the predefined weights of Kohonen network were used. The objective of this set of experiments was verification of the mapping creation of predefined world states (represented by neurons in Kohonen network) to actions. Various parameters of Q-learning (α , γ) were used. This part of experiments can be also divided into two parts: (1) adaptation of the single behaviours, when in each experiment only one behaviour was connected, achieving only one task/goal; (2) independent adaptation of both behaviours in the distributed manner, when both behaviours were connected and adapted by their selection controlled using subsumption network. The convergence of each behaviour could be observed approximately after 1000 iterations of learning algorithm in dependence of the behavioural activity. Convergence of the simple parking behaviour could be observed approximately after 3000 steps of simulation and the convergence of obstacle avoidance approximately after 1500 steps of simulation. This can be simply explained in terms of environment features. During robot-world interactions more avoid the walls and object situations occur than follow the attractor.

5.3 Combined perception and behaviour learning

In this set of experiments the Kohonen network learning (concept creation) as well as map learning was performed simultaneously. The purpose of this set of experiments was to find whether such a combination of 2 learning systems could find the desired behaviour and how long it takes compared with previous set of experiments. Kohonen learning took approximately 1000 learning steps to cover the input space and learning of good concept-action mapping took approximately another 5000 steps of simulation. Parameters for this experiments were selected from two previous sets of experiments.

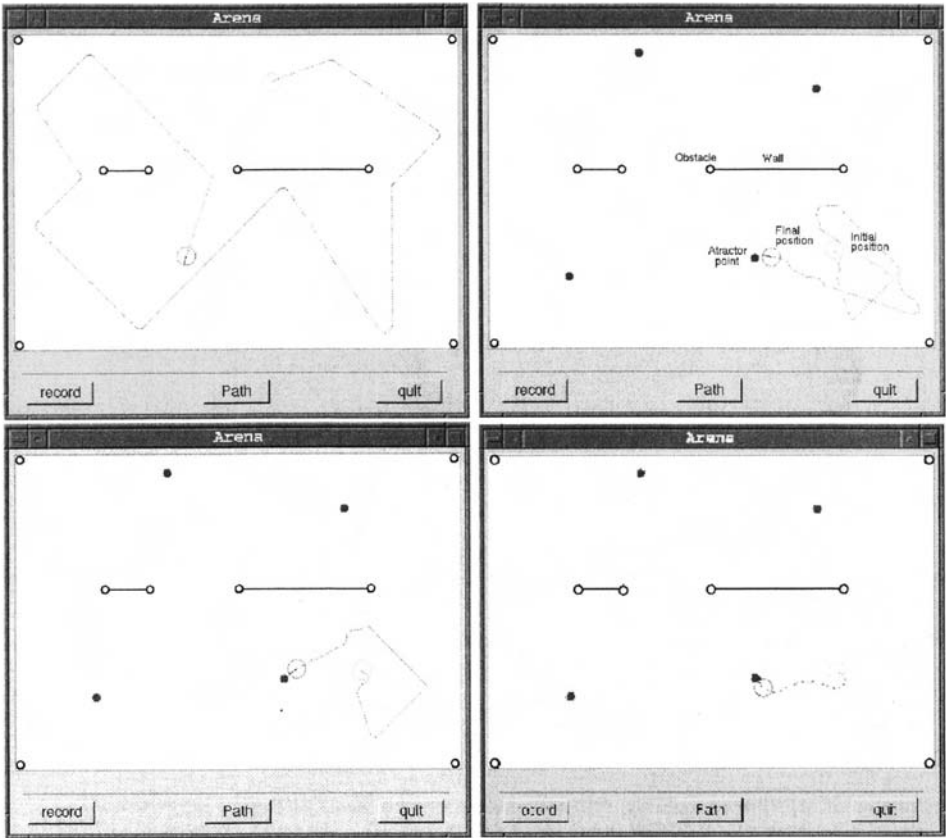


Figure 5: Examples: a) shows typical convergence result of obstacle avoiding behaviour; b), c), d) show evolution of control strategy using both behaviours.

Figure 5 shows typical examples of resulting behaviours. Figure 5a shows result of control strategy obtained in an experiment with adaptable perception network with used only obstacle avoidance behaviour after 2500 simulation steps. Figures 5b,c and d show emerging control strategy during an experiment with adaptable perception and both obstacle avoidance and parking behaviours connected. The experiment was focused on ability to achieve the attractor point. During the experiment three runs were realized with the same initial robot position and direction. The number of steps from initial position to attractor point was measured in each run. Results and convergence of control strategy for presented experiment are shown in Table 5.3.

Figure 5b shows control strategy after 500 simulation steps. The configuration of cluster structure and examples of assigned actions are shown in Figure 4a. Figure 5c shows control strategy after 2000 simulation steps where smooting of obstacle avoidance behaviour can be observed. Figure 5d shows control strategy after 4500 steps of simulation. The robot is able to reach the attractor point immediately. Representation of situation space and examples of action mapping are shown in Figure 4b.

Run No.	Iteration				
	Kohonen	Avoid	Find	Simulation	Steps to Goal
1	176	395	53	500	549
(52) 2	632	1241	393	2000	147
(366) 3	1655	2856	975	4500	16

Figure 6: Results of the experiment with three runs from the initial position to the attractor point.

6 Conclusion and further research

In this paper we presented an extension of our previous work, where we implemented learning behaviour based system on task of obstacle avoidance learning and simple parking problem. We implemented clustering algorithm learning of typical space states (sensory informations) and reinforcement learning for state-action mapping. The provided experiments shows that the Kohonen's learning algorithm succeeds on state (concepts) clustering with mostly equally distributed clusters. However there were some drawbacks of Kohonen network specially the need of explicit discretization of input space and either by random sampling of input patterns.

We performed several different experiments with aim to explore the stabilisation of control strategies. The desired (obstacle avoidance/parking) behaviour was usually learned after 1000 steps of learning algorithms in case of hardcoded situation concepts and in case of simultaneous learning of perception and behavioural modules it took approximately 5000 steps of simulation.

The further research will be focused on extension of new behaviour models and implementation of this architecture to the real LEGO robots. We also plan to analyse the influence of learning parameters on robot performance and learning time. The usefull extension seems to be substitution of the subsumption network with mechanism with adaptive behaviour arbitration.

References

- [1] R. S. Sutton, A. G. Barto, "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA, 1998
- [2] M. Gavalier, M. Hudec, R. Jakša, P. Šinčák: Computational Intelligence Controllers for Lego Robots - Comparison Study, Physica-Verlag, pp 288-295, 2001.
- [3] Kosko, B.: Neural Networks and Fuzzy Systems - A Dynamical Systems Approach to Machine Intelligence., Prentice-Hall International, pp 339-361 ,ISBN: 0-13-612334-1.
- [4] Maja J Mataric, "Learning in Behavior-Based Multi-Robot Systems: Policies, Models, and Other Agents", Cognitive Systems Research, special issue on Multi-disciplinary studies of multi-agent learning, Ron Sun, ed., 2(1), Apr 2001, 81-93
- [5] R.C. Arkin, Integrating behavioral, perceptual, and world knowledge in reactive navigation, Robotics and Autonomous Systems, vol.6, no.1-2, pp.105 122, 1990.
- [6] R.A. Brooks, Intelligence without representation, Artificial Intelligence, vol.47, no.1-3, pp.139 159, 1991.
- [7] P. Maes and R.A. Brooks, Learning to coordinate behaviors, Proceedings of the 8th National Conference on Artificial Intelligence, AAAI-90, pp.796 802, 1990.
- [8] C. Touzet :Neural Reinforcement Learning for Behaviour Synthesis, Robotics and Autonomous Systems, Special issue on Learning Robot: the New Wave, N. Sharkey Guest Editor, 1997.

Mass Approximation from Noise Signals of Loose Part Monitoring System

Š.FIGEDY, M.HREHUŠ

*VUJE Trnava, Inc.-Engineering, Design and Research Organization, Okružna 5,
91864 Trnava, Slovak Rep.*

Abstract. The purpose of this work is to show that the artificial neural networks with their pattern recognition ability can be applied in discriminating of noise signals generated by a loose part monitoring system. The wavelet transform was used to denoise the signals from the accelerometer sensors installed in a VVER-440 primary circuit. The data were then subject to FFT analysis and spectral features proposed in this work have been obtained. The discrimination of impacting part mass based on these spectral features was done by artificial neural network. The results show better performance of ANN than the classical approach using the spectral index. In the second phase the wavelet transform instead of FFT transform based signal features will be used as the inputs to ANN.

Keywords: *loose part monitoring system, noise, recognition, denoising, neural approximation, wavelets, FFT*

1. Introduction

The loose part monitoring system (LPMS) in the VVER-440 Dukovany or Bohunice NPP primary circuit (PC) as one of the fundamental diagnostic tools implemented is composed of a set of accelerometers mounted on the reactor vessel, main circulation pumps (MCP) and steam generators (SG), Figure 1. These sensors are connected to a processing unit. The purpose of the system is to detect and localize possible loose metal pieces and thus prevent the damage of PC components. Whenever the impacts are detected in the primary circuit the system is supposed to trigger an alarm and give an assessment of loose part mass and localization of impact.

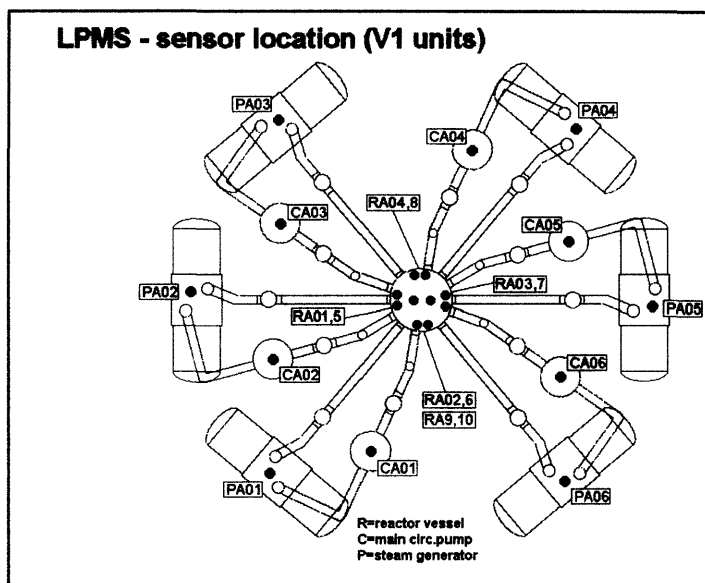


Figure 1. Layout of primary circuit components and LPMS sensors different impacting masses from the burst signals generated by the loose part monitoring system.

2. Noise cancellation

The loose part burst signal is a noisy signal especially when the point of impact is far from the sensor. The background noise generated by the MCPs and vibrations of the PC structures is superimposed and thus deteriorating the burst signals from LPMS sensors. In order the mass of the impacting part and the correct time of burst arrival for localization of impact might be determined, the signal has to be first cleaned off the noise.

The prerequisite of de-noising is that the real front rising edge of the burst should not be distorted, which can hardly be assured by the classical digital filters.

The burst signals were generated by a test impact hammer in different locations of the PC.

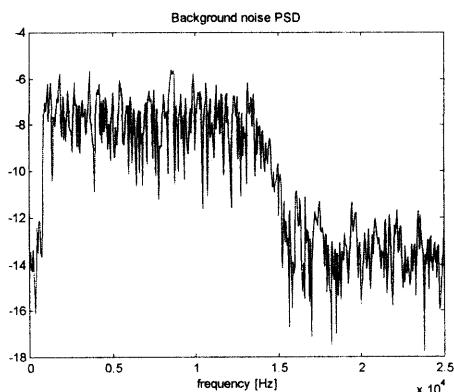


Figure 2. The background noise power spectral density

The signals consisted of 8192 samples taken by 50 kHz sampling rate. The power spectral density (PSD) of the background noise calculated from the initial part of the signal before the burst arrived is in **Chyba! Nenašiel sa žiaden zdroj odkazov..** It is apparent that the noise is a white noise the high frequencies of which are cut off by transmission characteristics of sensors and their fixing to PC structures. Frequencies below 1kHz have been removed by a high-pass filter.

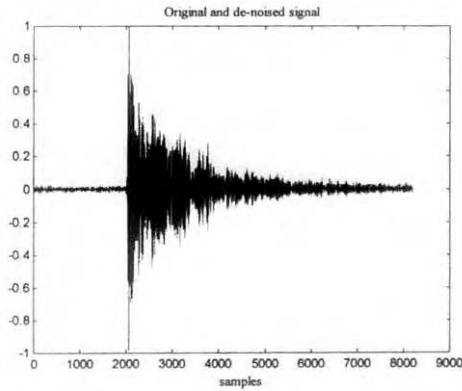


Figure 3. Wavelet transform based de-noising

For noise removing, the wavelet transform based de-noising has been applied. First the whole signal is decomposed up to the level five of details by MATLAB function *wavedec*, using Daubechies db2 wavelets. It is to say that the result of de- noising depends on the type of wavelets only insignificantly. Then the default soft threshold is calculated by the MATLAB *ddencmp* function only in the initial part of the signal before the burst arrived. Next the *wdencmp* function is called to threshold the wavelet coefficients from which the new signal is reconstructed. Figure 3. shows that the front rising edge of the burst is practically intact. This makes the wavelet based de-noising a good candidate for noise cancellation. On the other hand, we have found out that the FFT spectrum of de-noised signal unwanted distortion.

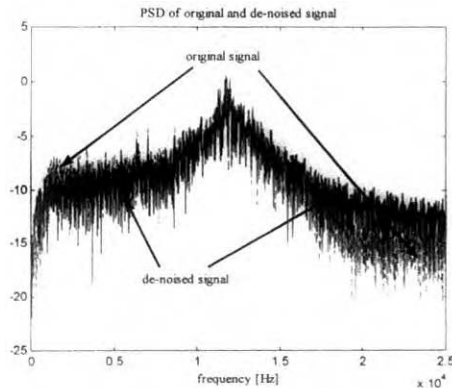


Figure 4. Spectral distortion after de-noising

In the lower frequency band the noise has been filtered out as expected while in the upper frequency band some new noise appeared, see Figure 4.

The explanation might be found in cutting off the peaks of all the spikes and thus adding more sharp edges to signal what is reflected by the high frequency tail in the FFT spectrum. In order this feature might be explained, more investigation will be needed. The above mentioned de-noising relates to the so called global thresholding when one threshold coefficient is calculated for all the scales. The other approaches of thresholding with threshold coefficient magnitude dependent on the respective scale have also been tested with little impact. Nevertheless, this distortion does not negatively influences the mass evaluation as frequencies only below 15 kHz were considered.

3. FFT spectrum as a function of loose part mass

The sensor response depends on the impacting loose part mass (and energy). This is also reflected in the shape of the burst FFT spectrum. The heavier the impacting part is, the more low frequencies can be found in the spectrum. In other words, the spectrum is shifted towards the low frequencies. In practice this dependence is expressed through the spectral index as a quantity characterizing the shape of the spectrum. The spectral index is the ratio of PSDs of low frequency band to high frequency band. In our case it was defined as

$$si = \sqrt{\frac{\sum_{1kHz}^{5kHz} PSD}{\sum_{5kHz}^{10kHz} PSD}}$$

The mass of the impacting part can be determined from a curve fitted to the spectral index values. This dependence on the mass is supposed to be “nice” monotonic, which is not always true in the real experimental practice as shown in Figure 5. This picture was obtained after averaging ten measurements of each mass from a sensor placed on the reactor vessel. The goal therefore was to find another more sensitive and unanimous method to determine the loose part mass from the spectrum.

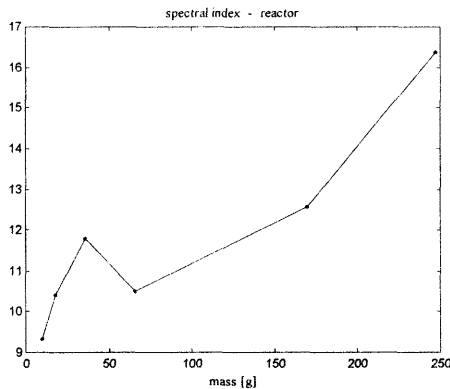


Figure 5. Spectral index dependence on the mass

4. Data pre-processing for neural approximator

We decided to use the artificial neural networks (ANN) trained to recognize the shape of the FFT spectrum. It is not feasible to bring the detailed spectrum to the input neurons

because then the ANN would be too large to be trained. For this task, features characterizing the spectrum have been defined. In the burst signal spectrum different resonance frequencies can be found. This is due to the transmission properties of structures, mechanical fixing of sensors and sensors themselves.

The change of spectrum due to the impacting mass is very much saturated in these resonancies and not in the rest of the spectrum. Having considered all types of spectra according to sensor localization and type of its fixing (magnetic or screw) and also the requirement for a low number of features (i.e. simple neural networks), we have found four frequency bands common for all spectra in which the mean values of PSDs represent the shape of the spectrum. Thus, four features defined as they follow characterize each spectrum:

$$sf_i = \frac{\ln(\overline{PSD_i})}{\ln(\overline{PSD_i})}, i = 1, 2, 3, 4$$

Here $(\overline{})$ represents the mean value.

Features constructed like this are relative numbers. Then, independently on the absolute values of PSDs the spectral features will always be in a relatively narrow range and only their mutual position in this range will characterize the shape of the spectrum. This is the good property for training the ANN. Figure 6 shows the dependence of spectral features on the mass of an impacting part. The solid lines connecting these features are only to enhance the clarity. Figure 7. shows the schematic view of ANN when just four features instead of a great number of inputs enter the ANN, which then calculates the mass of an impacting part.

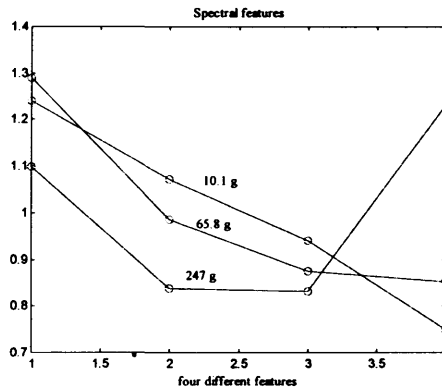


Figure 6. Spectral features dependence on the mass

To achieve this goal, the neural network has to be first trained to recognize the shape of the spectrum and assign the value of mass to it. The ANN can function as an approximator if in the training data spectra for different masses are represented. In our case burst signals for 10.1g, 18.4g, 33.6g, 65.8g, 170g and 247g were sampled. In the cross-validation training approach, few data from each mass were extracted from the training data. In addition to this, one complete data set for a specific mass was also extracted. As we had data from four NPP units, different mass for a particular unit was extracted. In the case shown in Figure 8 and Figure 9, it was data for 33.6g that were put aside. These extracted data unknown for the ANN were then used for testing how well the ANN was capable to

recognize the spectrum. The complete extraction of one data set representing one instance of mass enabled to test the ANN ability to approximate.

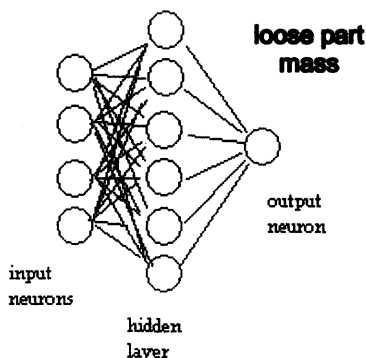


Figure 7. Schematic view of ANN

The topology of ANN depends on the number of inputs and outputs and character of data to be approximated. In our case we have four input neurons, each for one spectral feature, and one output neuron to give the estimation of mass.

Our task was to build up a neural approximator for which the output neuron was recommended to have a linear transfer function while the neurons in the hidden layer were supposed to have a sigmoid transfer function. What remained to set was the number of hidden neurons, which should be the lowest possible to prevent the network to memorize the data while at the same time to keep a good ability to generalize and thus to approximate with a desired accuracy. By trials and errors the number of hidden neurons was set to six as shown in Figure 7.

5. Results of neural approximation

The results of impacting part mass approximation on training data from a sensor placed on the reactor vessel are shown in Figure 8. The solid line represents the expected values of masses (as apparent, 33.6g line is missing from the training data), while the dots are the approximation of masses.

The results of approximation on testing data, i.e. on unknown data are shown in Figure 9. We can see that the approximation by neural network gives the acceptable accuracy. When we take into account that the classical approach based on spectral index failed the frequently, see Figure 5., then we are approved to say that ANN based loose part mass approximation has better performance.

6. Conclusions

The results of this work approve us to recommend the wavelet transform for LPMS signals de-noising. The determination of the time of burst arrival, i.e. the start of burst, which is usually done in linear prediction of a signal by predictive filters, will be after de-noising more accurate.

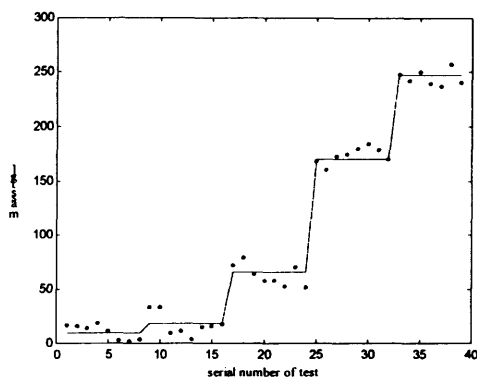


Figure 8. Approximation of mass on training data

Using of the artificial neural networks for loose part mass approximation seems to have better performance than the classical approach through the spectral indexes.

The work presented here is in its first phase when the aim is to show the capabilities of wavelet transform in signal de-noising and ANN in loose part mass approximation. The next plan is to investigate further their potential, e.g. in the more accurate estimation of burst signal time arrival, localization of impacting part and also to test the wavelet based instead of FFT based features for loose part mass approximation.

The reason for this is that we have found out that the FFT spectrum is not sensitive enough to the impacting mass – at least for the data we had at disposal.

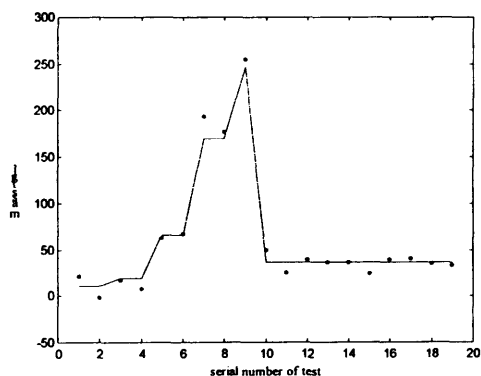


Figure 9. Approximation of mass on testing data

References

- [1] L.H.Tsoukalas, R.H.Uhrig, „Fuzzy and Neural Approaches in Engineering“, John Wiley & Sons, 1999
- [2] B. Kosko, „Neural Networks for Signal Processing“, Prentice-Hall International, Inc., 1992
- [3] P.E.Keller, „Nuclear Spectral Analysis via ANNs for Waste Handling“, IEEE Trans. on Nucl. Science, Vol.42, No.4, pp.709-715, 1995
- [4] G.Strang, and T.Nguyen, “Wavelets and Filter Banks”, Wellesley-Cambridge Press, 1996

Visualising Genetic Algorithms: A Way through the Labyrinth of Search Space

Marian Mach, Zuzana Zetaková

Department of Cybernetics and Artificial Intelligence, Technical University,

Letná 9, 041 20 Kosice, Slovak Republic

machm@tuke.sk

<http://neuron.tuke.sk/~machm/>

Abstract. Search space visualisation can increase user's understanding of processes behind the scene of a GA run and therefore foster better utilisation of possibilities provided by GA technology. The paper presents a simple and flexible method for visualising a high-dimensional search space based on projecting this space onto one dimension. The method provides users with opportunity to interact in order to adjust the way in which data are presented. A post-mortem GA visualising tool based on the proposed method is presented.

Keywords: *Genetic algorithm, Chromosome, Multidimensional data, Search space, Multidimensional scaling, Data projection, Data visualization*

1 Introduction

Software visualisation can be of different types but in domain of genetic algorithms (GA) there dominates visualisation of data produced as a result of a GA run (population data for each generation), although other types of visualisation techniques can be found as well, e.g. demonstration of crossover in operation [3] or tracing ancestral links between individuals or genes [5].

According to the level of granularity it is possible to distinguish three groups of visualisation techniques. These groups concentrate on visualising:

- population of chromosomes as a whole
- particular chromosomes
- alleles.

Visualisation of the whole population is very popular. Everyone involved in GA business is familiar with a simple Cartesian graph representing a fitness curve - fitness values over generations. This technique illustrates various fitness ratings (e.g. best, average, and worst fitness of a generation, online, offline fitness values) per generation. Another example is Distance Distribution Histogram aiming at population diversity [8].

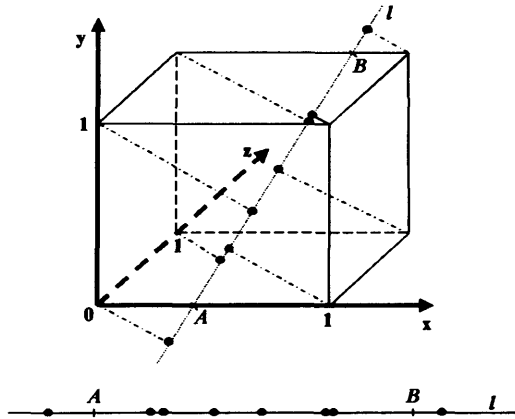


Figure 1: Projection of a high-dimensional data space onto one dimension. The position of line l represents a “viewpoint” which can be changed – it makes the presented approach quite flexible since it enables to view a distribution of chromosomes in different ways

The smallest elements used in visualisation of GA are alleles. Examples of methods from this category are Allele Fitness Quilt illustrating fitness of alleles [8], or distribution of alleles within a generation rendering internal structure of chromosomes [3].

A balance between the two extremes is represented by an approach focused on rendering each individual chromosome. It provides more information than an approach limited to only the whole population and at the same time it avoids unnecessary details about the structure of individuals. The remainder of this paper is devoted to this type of GA visualisation.

2 Projection-based visualisation of search space

It is very natural to visualise a (change of) distribution of chromosomes in search space represented by feature space of the used representation. This kind of GA visualisation enables to track such issues as generated path through the space, convergence of the population, coverage of the search space, etc. Unfortunately, the search space represents a high-dimensional data space. Since human observation capabilities are limited to three or less dimensions, all relevant visualisation methods are based on some way of transforming multidimensional data to a lower dimension. This transformation produces a low-dimensional picture of the search space. Typically, the data are transformed into a “screen space” – a plane, which can be displayed on computer monitors in an easy way.

The proposed approach is based on the placement of a line (called “target line”) in a high-dimensional search space and subsequent projecting each point representing a chromosome onto this line [12]. This process is illustrated in Fig. 1.

This transformation represents a simple mapping of linear computational complexity with respect to the number of chromosomes or the number of dimensions. In general, several different chromosomes can be projected into the same point on the target line although in case of binary representation it is possible to find such placement of the target line that there are no multiple projections.

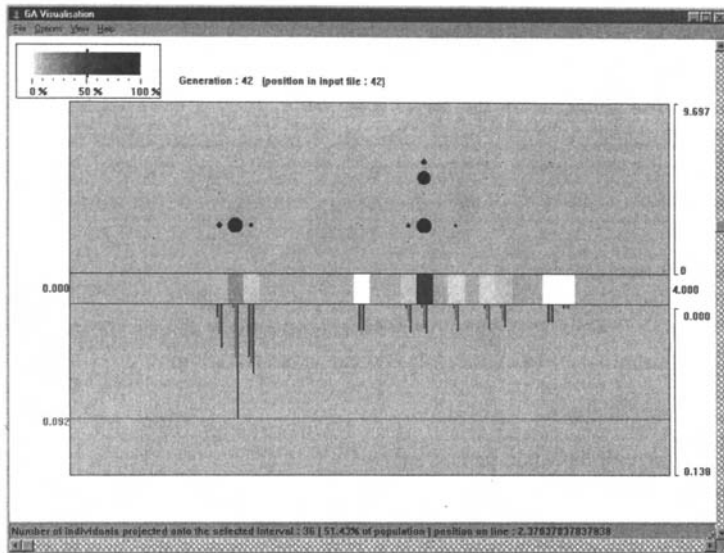


Figure 2: Screenshot of GA visualising tool. The window presents distribution of individuals in 2D (position on the target line against distance from this line), density of projected points on the target line, and worst, average and best fitness

Since one-dimensional visualisation is not so informative as using two dimensions (which is easy to follow for humans), it is possible to enrich the visualisation by adding one more dimension. The role of this can be played by distance between points to be projected and the target line.

3 GA visualising tool

The proposed method of visualising GA has been implemented in the form of an off-line visualising tool (its main screen is shown in Fig. 2) enabling to explore a GA run in a post-mortem way. Since the tool reads input data from a file, it can be used in connection with any GA software able to produce data describing a GA run in a proper format.

The tool enables us to define the position of the target line through co-ordinates of two points (unlike [11] where linear combination of axes is considered).

In addition to visualising only position of chromosomes in search space, some additional information is displayed as well. In order to express density in some area (due to projecting several chromosomes onto the same point of the target line or zoom out of the picture), it is possible to use points of different size (the larger point, the more chromosomes it represents) and replace the target line with a bar comprising rectangles of different colours (the darker colour, the more points are projected on this part of the line) positioned in the middle of the window. When some part of this bar is selected, then the scale in the upper left corner shows the percentage of population which is projected onto the given part of the target line. In addition, textual information is displayed in information line at the bottom of the window.

Fitness can be displayed in the bottom part of the window using a traditional method – as a histogram consisting of bars whose heights represent fitness values. Since several

chromosomes can be projected onto the same part of the target line, it is not possible to present fitness of all chromosomes separately. The solution is to present fitness of all chromosomes mapped onto the same part of the target line in the form of worst, average, and best fitness values related to the given part of the target line.

In order to distinguish among them, different types of fitness values can be represented by different colours. Moreover, adding a threshold representing the best fitness observed in previous generations enables to present progress in hunting for better solutions.

4 Experiments and results

Experiments with a few test problems proved the applicability of this approach as an alternative method for visualising GA. It can be used for visualisation of:

- coverage of search space
- convergence of population in search space
- evolution of distribution of chromosomes
- convergence on the allele level
- progress to better solutions.

The tool enables users to change their point of view. In order to adjust the way of visualising and to obtain the best picture of an evolving population, they can select:

- the position of the target line in search space
- a part of the target line which should be displayed (zoom).

In order to illustrate use of the visualising tool, let's consider a simple example - polynomial function approximation. The aim is to find coefficients of the following polynomial function

when given a set of tuples in the form $(x, f(x))$. It is possible to represent numbers as binary strings. Since we used five bits per variable, there are four variables (phenotype search space) or twenty (genotype search space).

Fig. 2 presents a view of 4D phenotype search space. Target line is defined by points (2,2,2,2) and (2,0,2,2) - it is parallel with the second axis. Thus, a projection of some chromosome onto the target line (the position of the projected point on the line) represents the value of the second variable of the given individual. The figure reveals two heavily populated subspaces defined by values 1.08 and 2.38. What is interesting is the fact that although the "left" subspace (1.08) is less densely populated than the "right" subspace (2.38) (17% of the whole population to 51%), the former has better fitness. Therefore it is natural to expect transfer of individuals from the "right" subspace to the "left".

Fig. 3 presents a view of 20D genotype search space. Since the target line is parallel with the 17th axis, the figure represents allele distribution for the 17th position (schemata of the first order). According to the figure, schema *****0*** dominates *****1*** (it can be found in more than 94% of the whole population).

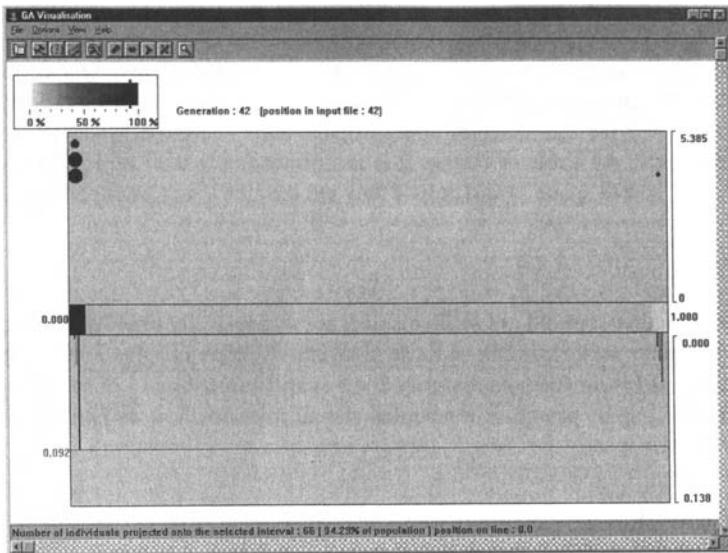


Figure 3: Visualising 2D search space. Target line parallel with the seventeenth axis

5 Discussion

The best way how to analyse the proposed method of visualisation is to compare it with other methods in use. Our method tries to visualise several information, but its core is the projection of points from high-dimensional space onto a line. Many methods for mapping multivariable data into low-dimensional space are currently known, e.g. Principal Component Analysis, Biplots [2], Sammon Mapping [4], Distance Maps [9], State Space Matrices [1], Coverage Maps [9], Correlation Tours and Grand Tours [11]. Unlike the proposed method, they typically provide mapping into two-dimensional space.

These methods have different characteristics. For some of them (e.g. biplots, principal component analysis, distance maps) there does not exist any common mapping for all generations but a new mapping is produced for each generation - these mappings reflect different distributions of individuals in different generations. Thus, there is no consistent spatial relationship between the plotted points of subsequent generations - the same individual can be mapped into different points in different generations. Another issue is computational complexity, which makes some methods intractable (or at least impractical) for real problems. As an example can serve Sammon mapping the complexity of which is quadratic with respect to the number of data points although there exist a few modifications decreasing the complexity [6]. Some of the presented methods are not suitable for floating point number representations (e.g. state space matrices). These methods can be used to visualise only genotype search space, which is less meaningful for users than visualising in phenotype search space.

The proposed method has both advantages and disadvantages. As an advantage it guarantees the same mapping for each generation. Thus, it is easy to follow a chromosome through generations. Moreover, it is possible to visualise chromosomes in both genotype and phenotype search spaces. In order to change the point of view, user can change the position of the line. In addition to visualising position of chromosomes, special positioning of the target line

enables to explore distribution of alleles for a locus and therefore it is possible to make conclusions about convergence of the population on allele level. The computational complexity of the mapping is linear in the number of individuals and the number of dimensions.

As a disadvantage can be presented the fact that it is not obvious which position of the target line is optimal for visualisation of a particular run. Therefore it could be difficult to choose the right one. As a rule of thumb, it is recommended to use "typical" positions (e.g. "along an axis", "across space") and only if they are unsatisfactory to experiment with it.

6 Conclusions

Visualising GA is almost as old as GA themselves for its capability to provide GA developers and users with better understanding of these algorithms' behaviour. In spite of this fact, the research in this field is far from its maturity and has still something to offer. The aim of this paper is to fill this gap by providing a practical visualisation method for exploring the search path through search space.

Acknowledgements

We would like to thank an anonymous reviewer for helpful comments and hints for further work. The first author gratefully acknowledges the support of VEGA grant 1/8135/01 of Scientific Grant Agency of Ministry of Education of the Slovak Republic.

References

- [1] Collins, T.D.: Understanding Evolutionary Computing: A Hands on Approach. In: Proc. of the International Conference on Evolutionary Computation ICEC'98. Anchorage, Alaska (1998)
- [2] Collins, T.D.: Using Software Visualization Technology to Help Evolutionary Algorithm Users Validate their Solutions. In: Proc. of the 7th International Conference on Genetic Algorithms ICGA'97. Michigan (1997) 307-314
- [3] Durie, R.: VisualGA: Visualisation of a Genetic Algorithm. Report EPCC-SS98-05. Edinburgh Parallel Computing Centre, University of Edinburgh, UK (1998)
- [4] Dybowski, R., Collins, T.D., Weller, P.R.: Visualisation of Binary String Convergence by Sammon Mapping. In: Proc. of the Fifth Annual Conference on Evolutionary Programming EP'96. San Diego, CA (1996) 377-383
- [5] Hart, E., Ross, P.: Enhancing the Performance of a GA through Visualisation. In: Proc. of the Genetic and Evolutionary Computation Conference GECCO-2000. Las Vegas, Morgan Kaufman (2000).
- [6] Pekalska, E. et al.: A new method of generalizing Sammon mapping with application to algorithm speed-up. In: Proc. of the 5th Annual Conference of the Advanced School for Computing and Imaging ASCI'99. Delft (1999) 221-228
- [7] Pohlheim, H.: Visualization of Evolutionary Algorithms - Set of Standard Techniques and Multidimensional Visualization. In: Proc. of the Genetic and Evolutionary Computation Conference GECCO'99. Orlando, Florida (1999) 533-540
- [8] Routen, T.: Techniques for the Visualisation of Genetic Algorithms. In: Proc. of the First IEEE Conference on Evolutionary Computation. Piscataway, New Jersey (1993) 846-851
- [9] Shine, W.B., Eick, C.F.: Visualizing the Evolution of Genetic Algorithm Search Processes. In: Proc. of the International Conference on Evolutionary Computation ICEC'97. Indianapolis (1997) 367-372

- [10] Spears, W.M.: An Overview of Multidimensional Visualization Techniques. In: Evolutionary Computation Visualization Workshop, GECCO'99 Bird-of-a-feather workshop (1999)
- [11] Swayne, D.F., Cook, D., Buja, A.: XGobi: Interactive Dynamic Data Visualization in the X Window System. *Journal of Computational and Graphical Statistics* 7 (1998) no. 1.
- [12] Zetkova, Z.: Visualising Activities of Evolutionary Algorithms [in Slovak]. MSc. Thesis. Dept. Of Cybernetics and Artificial Intelligence, Technical University in Kosice, SR (2001)

Genetic Algorithm Optimization of the Dataflow Processor

Michal TURČANÍK, Miroslav LÍŠKA

*Dept. of Informatics and Computer Sciences, Faculty of Telecommunication Technology
 and Information Systems, Military Academy in Liptovský Mikuláš, Slovak Republic*
 turcanik@valm.sk, liska@valm.sk

Abstract. The paper deals with the use of genetic algorithms for structural optimization of the dataflow processor. The goal of the optimization is to obtain the fittest structure of the dataflow processor for execution of a specific program. Structural optimization of the dataflow processor is possible to do by operation units optimization and data queue sizes optimization. The applicability of the structural optimization of the dataflow processor by genetic algorithms was analyzed for several algorithms.

Keywords: *dataflow processor, genetic algorithms, operation unit, data queue*

1. Introduction

Activity of a dataflow processor (DFP) is controlled by data [1, 2, 3, 4]. Computational operations are processed by operational units configured on-the-fly thanks to the reconfigurable processor structure. Demand on capacity of reconfigurable operational unit (ROU) space is usually higher than available hardware resources [5]. The ROU space is realized by field programmable gate arrays (FPGA) that are locally configurable [6, 7]. The architecture of the dataflow processor, characteristics of function units, and description of processor activity are presented in [8, 9]. Genetic algorithms (GA) model natural evolution. They can be utilised for optimization of tasks. Optimization using GA utilizes problem encoding to the string of parameters (also denoted as chromosomes or genotypes) representing the optimized system. The first population of chromosomes is generated randomly. Subsequent generations are obtained by combining the best chromosomes evaluated according to the application specific fitness criteria. The most common combination procedure is crossover. It combines two chromosomes to generate a new one. Another essential operation in GA is mutation, which randomly modifies chromosome genes. This preserves diversity in the solution space and assures escape mechanism from local optima. The genetic algorithms work at abstracted level. Problem encoding defines the available search space for GA. Genetic algorithms are strictly guided by fitness criteria. It makes GA effective for multi-dimensional optimization. They have been successfully employed in variety of applications [10, 11, 12].

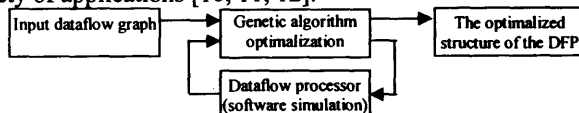


Figure 1 – Block scheme of the optimization of the dataflow processor

2. The optimization of the dataflow processor

Capacity constraints of the dataflow processor are given by ROU space, the number and the size of data queues. The operations of the program are realized by operation units. The operation units are configured in the ROU space, whose capacity is limited. The data queues are used to store data during the program execution. The number of the data queues is limited too. Due to these limitations the structural optimization of the dataflow processor is necessary. The result of the optimization is the structure of the dataflow processor having the shortest execution time for the given program. The activity of the dataflow processor cannot be described by function suitable for standard optimization methods (linear or nonlinear programming, steepest descent or ascent optimization, etc.). The dataflow processor activity is described by a state machine for each function unit. This type of dataflow processor activity description is suitable for creating of the simulation model. The determination of the number of clock cycles for program execution can be done by the simulation model of the dataflow processor. The number of clock cycles for program execution is sufficient information about specific structure of the dataflow processor for genetic algorithms (Figure 1). It is possible to do structural optimization of the dataflow processor by:

- operation units optimization followed by data queue sizes optimization,
- operation units and data queue sizes optimization at the same time.

2.1 The optimization of the operation units of the dataflow processor

The input of optimization is a set of operations representing the dataflow graph for a given program. Some operations from this set can be processed by various structures. For example: the operation of addition can be realized by a serial or parallel adder. The difference between these two adders is the execution time, configuration time of FPGA and space required in the ROU part of the processor. From the input set of operations the special operations (adder, multiplier, etc.) are chosen for GA optimization. Each special operation is represented by one gene of a genotype that has as many alleles as the number of structures for its possible physical realization. The special operations define the genotype. The creating of the genotype for operation units optimization is shown in Figure 2. Specific genotype represents one particular solution of an optimization problem. The fitness criterion for the genotype is the execution time of the program and data under consideration. The output of the GA optimization is a selected set of operations, such that the dataflow processor executes the program in the shortest time. This set of operations represents the fittest structure of the dataflow processor for the given program.

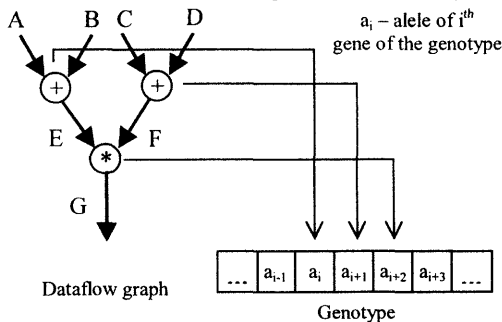


Figure 2 – The operation units optimization of the dataflow processor

2.2 The optimization of the data queue size of the dataflow processor

The data queue size optimization of the dataflow processor follows the operation units optimization. It means that input to the optimization is a structure of the dataflow processor with optimized operation units. For each operand of input set of operations one gene of the genotype is reserved. This gene represents the size of data queue, which will be associated with the given operand in program execution. The creating of the genotype for data queue sizes optimization is shown in the figure 3. The creating of the genotype for data queue sizes optimization compared with genotype creating for operation units optimization uses identifiers of the operands, which is realized by data queue, not identifiers of the special operations. The specific genotype represents one particular solution of the optimization problem. The fitness criterion for the genotype is the execution time of the program and data under consideration. The output of the GA optimization is a set of the data queue sizes that the dataflow processor executes the program in the shortest time. This set of the data queue sizes represents the fittest structure of the dataflow processor for the given program.

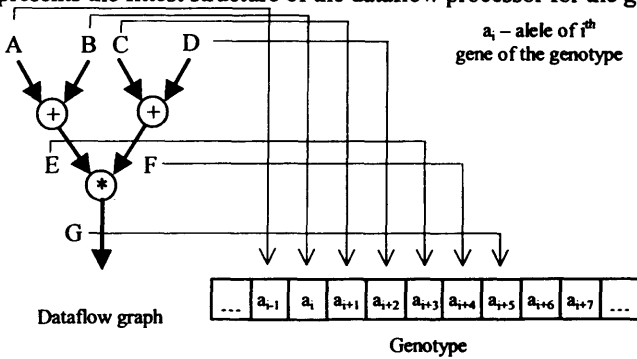


Figure 3 – The data queue size optimization of the dataflow processor

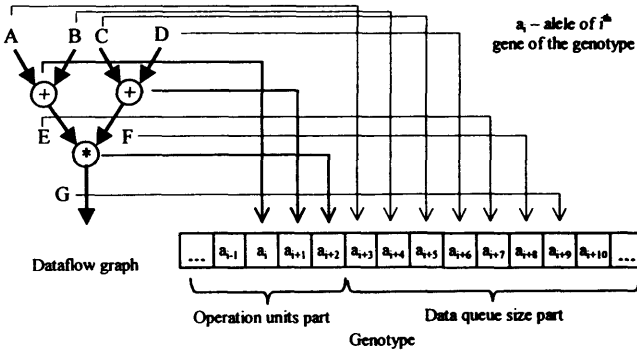


Figure 4 – The operation units and data queue size optimization of the dataflow processor

2.3 The optimization of the operation units and data queue size of the dataflow processor

The input of optimization is a set of operations and a set of data queues representing the dataflow graph for a given program. Special operations define the first part of the genotype. For each operand of the input set of operations one gene of the genotype is reserved. This gene represents the size of the data queue, which will be associated with a given operand in program execution. The second part of the genotype represents the size of data queues (Figure 4). The creating of the genotype for operation units and data queue sizes optimization compared with genotype creating for previous manners of the dataflow

processor structure optimization uses identifiers of the special operations and identifiers of the operands together, not identifiers of the operation units and operands individual. A specific genotype represents one particular solution of optimization problem. The fitness criterion for the genotype is the execution time of the program and data under consideration. The output of the GA optimization is a selected set of operations and the size of the data queues, such that the dataflow processor executes the program in the shortest time. This set of operations and the set of the data queue sizes represent the fittest structure of the dataflow processor for the given program.

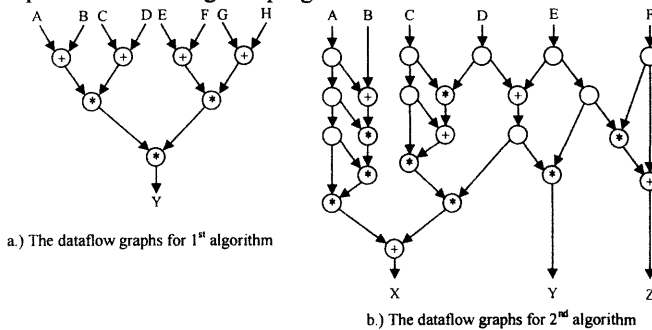


Figure 5 – The dataflow graphs for analyzed algorithms

3. The analyzed algorithms

The applicability of the structural GA optimization was analyzed for three algorithms. The first and the second algorithms were randomly generated (Figure 5). The last analyzed algorithm is the encryption algorithm RSA. The first algorithm has seven special operations and 15 data queues. The second algorithm has 13 special operations and 34 data queues. The RSA algorithm has 15 special operations and 190 data queues. For each algorithm the structure of the dataflow processor was optimized and the results were compared. It is possible to do structural optimization of the dataflow processor for analyzed algorithms by operation units optimization and data queue sizes optimization.

4. The conclusion

The results show the suitability of genetic algorithms to solve the task of structural optimization of the dataflow processor. The use of genetic algorithms decreases the time for finding the suboptimal structure of the dataflow processor. The comparison of the results of the operation units optimization, data queue sizes optimization and operation units and data queue sizes optimization at the same time for analyzed algorithms is shown in Figure 6. The operation units optimization of the dataflow processor reduces the time which is needed for program execution. The operation units optimization followed by the data queue sizes optimization can bring additional reduction of the execution time of the dataflow processor for analyzed algorithms. The results of the operation units and the data queue sizes optimization at the same time are comparable with the results of the operation units optimization which is followed by the data queue sizes optimization. The operation units and the data queue sizes optimization at the same time have lower time complexity than the operation units optimization followed by data queue sizes optimization. The advantage of using of genetic algorithms for the dataflow processor optimization is a possibility of the adaptation of the dataflow processor structure to the defined algorithm. The disadvantage of using of genetic algorithms is high time complexity.

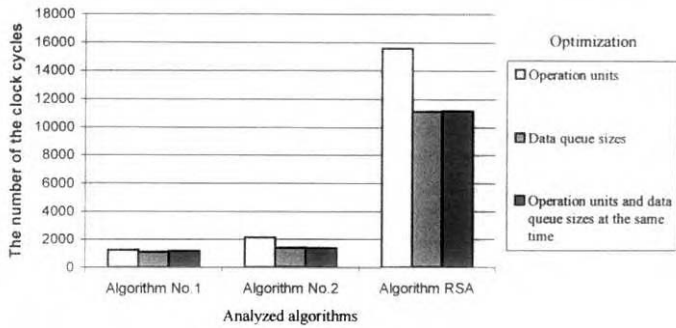


Figure 6 - The comparison of the results of the analyzed manners of the DFP optimization

References

- [1] Giraud-Carrier, C.: A Reconfigurable Data Flow Machine for Implementing Functional Programming Languages. SIGPLAN Notices, 29(9), September 1994, pp. 22-28.
- [2] Kienhuis, A. C. J.: Design Space Exploration of Stream-based Dataflow Architectures. PhD thesis, Technical University Delft, 1999.
- [3] Williamson, M.: Synthesis of Parallel Hardware Implementations from Synchronous Dataflow Graph Specifications. Ph.D. thesis, Memorandum UCB/ERL M98/45, Electronics Research Laboratory, University of California, Berkeley, May 1998.
- [4] Xiao, P. L. – Hideharu, A.: An MPLD with Data-Driven Control on Virtual Hardware. The Journal of supercomputing, Number 9, 1995, pp. 253-276.
- [5] Xilinx, Inc., The Programmable Logic Data Book, San José, CA 95124, January 2001.
- [6] Altera Corporation, Device Data Book, San José, CA95134, May 2001.
- [7] Wirthlin, M. J. – Hutchings, B. L.: Improving Functional Density through Run-Time Constant Propagation. ACM/SIGDA International Symposium on Field Programable Gate Arrays, Monterey, CA, 1997, pp. 86-92.
- [8] Turčaník, M. – Liška, M.: Possible approach to design a dataflow processor using reconfigurable logical circuits. In Proceedings of New Trends in Signal Processing V, Liptovský Mikuláš, May 2000, pp. 378-385 (in Slovak).
- [9] Turčaník, M. – Liška, M.: The dataflow processor and reconfigurable logic. In Proceedings of Conference SECON '99, Warsaw, November 1999, Poland.
- [10] Torbey, E. – Knight, J.: Performing Scheduling and Storage Optimization Simultaneously Using Genetic Algorithms. 11th Annual IEEE International ASIC Conference, August 1998, Canada.
- [11] Cluitmans, L.J.M.: Using Genetic Algorithms for Scheduling Data Flow Graphs. EUT Report 92-E-266, December 1992, Holland, pp. 21-52.
- [12] Koza, J. R. – Bennett, F. H. – Andre, D. – Keane, M. A.: Evolutionary Design of Electrical Circuits Using Genetic Programming. Proceedings of Adaptive Computing in Design and Manufacture conference, Plymouth, England, April 1998, pp. 159-174.

Pareto-Optimization with EA in Application of Centrifuges Design

Xue FENG and Jens STRACKELJAN

Technical University of Clausthal, Institute of Applied Mechanics
38678 Clausthal-Zellerfeld, Graupenstr. 3, Germany

{xue.feng jens.strackeljan}@tu-clausthal.de

Abstract. Applications for evolutionary algorithms are growing even in those areas of machine construction hitherto reserved exclusively for model supported simulation programs. By way of example, this article shows how an evolutionary algorithm with Pareto optimality can help solve a longstanding problem related to optimizing the design of modern laboratory centrifuges. Of course, similar applications are conceivable for a wide range of other designs.

1. Introduction

In the design of high powered laboratory centrifuges, security and comfort of machines have to be considered from the beginning of the design process. Centrifuges, depending on the practical application, may have a weight from 40 kg to 400 kg with a speed of between 15,000 to 30,000 rpm. Therefore, the dynamic behavior for the machine group with such high speeds is the most important design criterion.

As shown in Fig. 1 the shaft connects the drive motor and the rotor and often takes the form of a thin needle, thus preventing the strong bending stresses. A foot mass element has been attached to the bottom end of the motor to increase the mass of the motor, and the arrows are intended to indicate how changing the place of this mass naturally changes the position of the center of gravity and the axial inertia of the motors. This foot mass is often devised as a solid design of the motor mount shield because many designers hope this will result in smoother running of the machine. Without going into the technical details of the machine dynamics, damping, speed-dependent natural frequencies of the rotor system, excursions and stress of the shaft are representative of the dynamic behavior according to the design guidelines [Strackeljan]. The design of this model appears relatively simple, and a suitable program system can be used to calculate the required parameters of dynamic behavior.

Modern laboratory centrifuges are characterized as being designed like “universal machines”, so a wide variety of different rotors can run on it. On the other hand the spring suspension elements, the entire drive and the shaft connecting the motor and rotor are the same all the time. Obviously the system behavior is affected if the essential inputs like mass and inertia are changed. In this case we might speak of an entirely different machine.

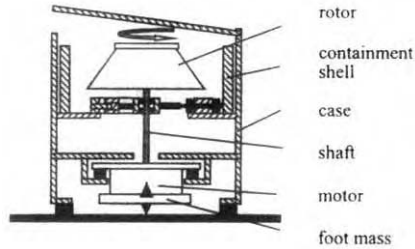


Figure 1. Schematic representation of laboratory centrifuges

For the engineering design it is helpful to yield optimal solutions if the input parameters can be varied within prescribed, technically reasonable limits in the simulation program. However, most calculation programs for machine dynamics are lacking an optimization module. The results presented here were calculated by the LUBEST [Lubest] simulation program developed at the Institute of Applied Mechanics and has been used by many centrifuge manufacturers in Germany for about ten years, although it too does not have a module for optimization, too.

2. Evolutionary Algorithm for Optimization

Before the machine layout in the simulation program can be accessed, more than 20-30 system parameters have to be specified before even a simple calculation can be carried out. (A selection of the relevant system parameters and the corresponding values for the centrifuges presented are listed in tab 1.) Noteworthy is the variability of the mentioned inputs, and so choosing the parameter values will leave the designer much scope for development, but it is also a matter of finding good compromises. Since an optimal solution is the basic requirement of the general system behavior, this means that a combination of damping, speed-dependent natural frequencies, excursions, stress and optimization of centrifuges is a multiobjective optimization problem.

The complexity of this effort at optimization becomes eminently clear when consideration must be given to additional factors such as bearing forces, the position of the frequencies, lateral forces in the shaft which also contribute to the overall tension in the shaft, etc. It then becomes totally impossible to do a systematic screening of the range of possible solutions, and so powerful search methods have to be used.

Relevant system parameters	Values
Rotational speed	150 to 300 Hz
Shaft diameter	0.02 m
mass motor	15 kg
mass rotor	20 kg
ax. Moment of inertia (motor)	0.05 kg m ²
ax. Moment of inertia (rotor)	0.3 kg m ²
polar moment of inertia (motor)	0.5 kg m ²
foot mass	0.0 to 5 kg

Table 1. Main system parameters of the present design optimization

Evolutionary Algorithms (EA) have proved to be effective methods of optimization in the past, and since the authors have been able to gather experience with this method in connection with the selection of features in classification problems, naturally the choice here for optimization is EA. The basic algorithm can be summarized by the following pseudo code.

EA:

```

t := 0
Initialize (P0)
evaluate (P0)
do while not terminate
    P't := variation (Pt)
    evaluate (P't)
    Pt+1 := select (Pt ∪ P't)
    t := t+1
enddo

```

In our case, we started with a randomly produced parent generation whose offspring produce new solutions spanning several generations determined by a weighted average of two parents (crossover) and then randomly changing the offspring (mutation). To evaluate the solutions, the factors to be optimized are gathered together in performance functions in the fitness function. The next step is seeking the best individuals to be adapted from the next generation.

In single-objective problems the individuals (children) are normally selected according to the expectation of fitness function into a feasible set of individuals. Instead of this at the multiobjective optimization, the distinguished criteria are simultaneously involved and a selected individual can not be improved in any objective without degradation in another objective. Therefore, it means selection is dependent on the priority of the optimization objective, which must be decided. A solution to this problem is a fixed optimization rank. Therefore, an objective with high standing rank at optimization is intended to be realized first, with any other possible proposed compromises at optimization rejected. Contrary, in machine design the optimization ranks of several objectives are generally flexible. Preferable is getting a feasible set to satisfy the objectives whose ranks are optionally combined.

3. Pareto Optimality for Multiobjective Optimization

The concept of Pareto (P) optimality similar to Goldberg's "does not help us to select a single alternative from the P-optimal set. The decision maker must ultimately make a value judgement among the alternatives to arrive at a particular decision." [Goldberg]

The conditions of Pareto optimality are mathematically presented in Eq.1. A vector \underline{m} is partially (assigned as p) less than \underline{n} , symbolically $\underline{m} <^p \underline{n}$, when it obtains

Vector \underline{m} is Pareto-optimal (P-optimal) in the sense that no element in the vector \underline{n} exists to dominate the vector \underline{m} .

Fig. 2 shows a principal example of Pareto optimality. Evidently here is $D > B$ and $B > A$, but there is no judgement between D and C, because in the fitness function 1 D is in fact better than C, but worse than C in the fitness function 2. On examination of Eq. 1, the elements in the P-optimal front (P-front) are unique among A, B, C, D because they improve at least one of the fitness functions better than A, B, C and D.

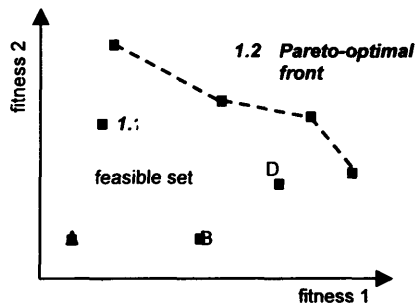


Figure 2. Principal scheme to Pareto Optimality and Pareto-Front (P-front)

In the selection technique with Pareto optimality it is observed that the samples of P-front are concentrated at certain areas and the stochastic characteristic of EA cannot be performed, when only front solutions are selected as candidates of the next generation. The optimal solutions then found might be finally local optimum. To maintain the variability of the chosen individuals it is effective to add some individuals, which are not part of the front.

In the following example of a laboratory centrifuge (Fig.1) we optimized the bending stress of the shaft (fitness 1) and the angle torsion of the rotor (fitness 2). The value 1 in the single-objective fitness functions presented in the Figures 3 and 4 are here defined for the minimal stress and the minimal angle torsion. The aggregated fitness function is presented in Fig. 5 to facilitate comparison with the P-optimal solutions. It is calculated by using the equally weighted single fitness functions 1 and 2.

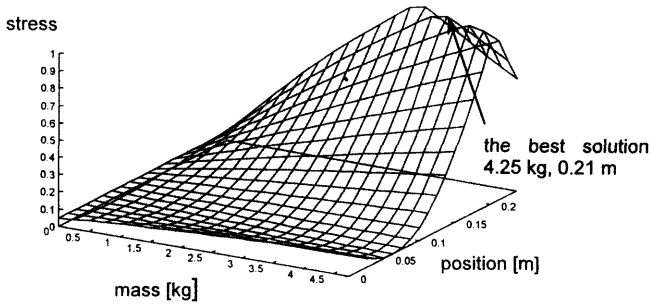


Figure 3. Bending stress of shaft as fitness function of foot mass and position

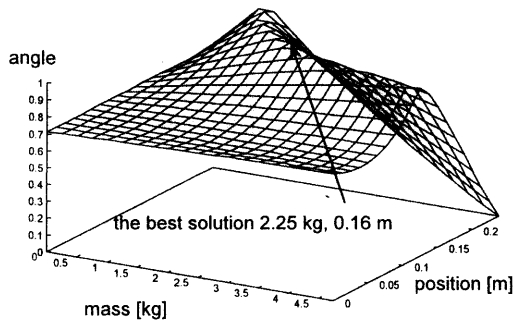


Figure 4. Angle torsion of rotor as fitness function of foot mass and position

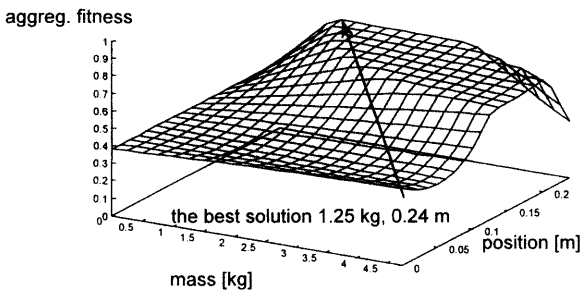


Figure 5. Aggregated fitness function of foot mass and position as average of angle torsion of rotor and bending stress of shaft

Fig. 3 shows the application of the relative stress in the drive shaft in relation to the foot mass and the position where it is attached on the motor. Within the prescribed system sizes, a mass of 4.25 kg yields the least stress when it is attached 0.21 meters from the lower edge of the motor. It is also possible to design significantly poorer centrifuges with considerably higher stresses simply by moving the mass upward or downward slightly. In addition if you

consider the angular torsion of the rotor, which should also be kept as small as possible to prevent the rotor from scraping in the containment shell, in opposition to the stress factor mentioned above, then a mass of 2.25 kg and a position of 0.16 m prove to be the minimum as shown in Fig. 4.

It is of great advantage to use P-front since the weighting of the fitness functions do not have to be determined at the beginning of the optimization. At the end of EA the P-front of the last generation can be used as a basis group for the optimal design option.

Fig. 6 shows the samples of four P-fronts during the selection process. In the first generation only three front values could be determined as P-optimal solutions, the number of front values increasing with each generation. For this example 30 generations with a population number of 50 were calculated. The improvements are visible from the 10th generation, however the quality of the solutions increased only slightly. There are also solutions (e.g. $F_{\text{stress}} = 0.96$ and $F_{\text{angle}} = 0.4$) which are in the 10th Generation still on the P-front, but in the 30th Generation are disappeared. When it is intended to keep a good solution unchanged which has been found once in a previous generation, an elite selection (elitism selection) could be included for these individuals who have survived without mutation for arbitrary generations. Since the individuals of P-fronts depend strongly on the quality of the other solutions to the same population, the elitism selection is not used.

The best solution of the aggregated fitness function with a mass of 1.25 kg and a position of 0.23 m achieve values of $F_{\text{angle}} = 0.47$ and $F_{\text{stress}} = 0.9$ in the individual fitness functions. Exactly this solution is appeared in the P-front of the 30th generation. Obviously the solution of the aggregated fitness function is part of the P-front

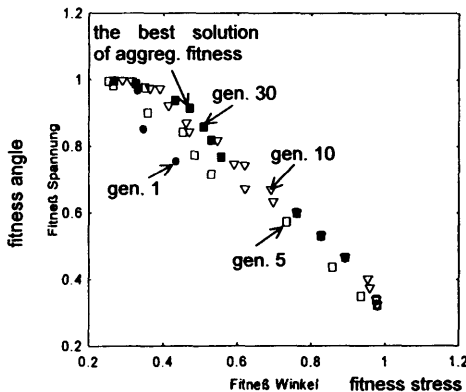


Figure 6. Pareto fronts for 4 generations.

4. Summary

Finding optimal solutions of multiobjective problems by using numerical simulation programs systematically to create different designs will only be successful in isolated cases, and it will no longer be possible to do so without powerful optimization modules in the future. Discovering good solutions using evolutionary algorithms with Pareto optimality is

a very promising technique and is also quite likely to find more widespread use in simulation-supported design in the future.

References

- [1] Bäck, Th.: Principles of Evolutionary Computation. Proc. EUFIT 97 Third European Congress on Intelligent Techniques and Soft Computing, Aachen, Germany, pp. 637-639.
- [2] Bäck, Th.: Evolutionary Computation: History and Current State. IEEE Trans On Evolutionary Computation 1(1) pp. 3-17, 1997.
- [3] Dasgupta D. u. Michalewicz (Hrsg.), Z.: Evolutionary Algorithms in Engineering Applications, Springer, Berlin, 1997.
- [4] Fonseca, C. M. a. P. J. Fleming: An Overview of Evolutionary Algorithms in Multiobjective Optimization, Evolutionary Computation 3 (1), 1-16, 1995.
- [5] Gasch, R.; Pfützner, H.: Rotordynamik. Springer Verlag 1975.
- [6] Goldberg, D. E.: Genetic Algorithm in Search, Optimization, and Machine Learning. Addison-Wesley 1989.
- [7] Klement, H. D.: MADYN 4.0, Programmsystem zur Maschinendynamik, Darmstadt, 1994.
- [8] Lubest: Programmsystem zur Berechnung von Laborzentrifugen, Technical University of Clausthal, Clausthal, 1991.
- [9] Pareto, V.: Cours D' Economie Politique, Volume I, Lausanne, F. Rouge, 1896.
- [10] Strackeljan, J., Behr, D. u. Damm, A.: Untersuchungen zur Struktur Periodischer Lösungen von Rotoren mit Anisotrop Nichtlinearer Lagerung, Schwingungen in rotierenden Maschinen IV, Vieweg Verlag, 1997.
- [11] Strackeljan, J.: Einsatzmöglichkeiten von Softcomputing-Methoden zur Auslegung, Optimierung und Überwachung von Rotorsystemen. Habilitationsschrift TU Clausthal, Clausthal 2002.
- [12] Yang Y., Wang Z.: Analysis of the Dynamic Characteristics of Lab-Centrifuges and its Application, Proc. 1th Conference on Stability Control of Rotating Machinery, Lake South Tahoe, 2001.
- [13] Zhao, C., Steven G. P., Xie, Y. M.: General Evolutionary Path for Fundamental Natural Frequencies of Structural Vibration Problems: Towards optimum from below , In: Structural engineering and mechanics , 5, S. 513-528, 1996.
- [14] Zitzler, E.: Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications. Swiss Federal Institute of Technology Zurich, Zurich 2000.

Neuro-Tomography: A New Approach to Image Reconstruction from Projection

Robert CIERNIAK

Technical University of Częstochowa

Department of Computer Engineering

42-200 Częstochowa, ul. Armii Krajowej 36

Poland

ciemniak@kik.pcz.czyst.pl

Abstract. This work represents a new approach to a well-known problem: image reconstruction from projections. The most important application of this technique is computer tomography — one of the basic diagnostic methods in medicine. Presented solution of this problem uses autoassociative neural network and maximum entropy criterion.

Keywords: *image reconstruction from projections, neural networks, entropy.*

1. Introduction and problem definition

Computer tomography, especially its algorithmic aspect, is still a very rewarding field of investigations. Since Cormack's publication [1], one of the key tasks in the field has been to integrate into the studies many new algorithms and apply them to the task of reconstructing an image from projections. The most important reconstruction methods are the ones using convolution and back propagation [3], the invert Fourier transformation and an algebraic reconstruction technique (ART) [3-4]. Considering the increasing amount of soft computing algorithms applicable to different science disciplines, it is possible that in the foreseeable future these algorithms will occupy an important place in computer tomography. This work represents an attempt to approach image reconstruction from projections by using neural networks — the most popular and important tool of artificial intelligence systems. Presented in such an application, the definition of a neural network is based on the maximum entropy criterion [2-5], which meets with approval among scientists representing different fields of knowledge. Advantages of this definition have been demonstrated during computer simulations designed to prove the assumption that an appropriately constructed neural network is able to reconstruct an image using projections.

1.1 Formulation of the problem

In medical diagnostics knowledge about distribution of affiliation in the investigated body gives extremely useful information about both the tissue layers and any pathological changes. Fig. 1 shows a single projection using x-rays collimated into a parallel beam of very small thickness. The function below gives the distribution of radiation affiliation reaching the screen when a projection is made at a specific angle. It is called the Radon's transformation [6]:

$$p(s, \alpha) = \int_{-\infty-\infty}^{\infty \infty} \int_{-\infty-\infty}^{\infty \infty} \mu(x, y) \delta(x \cos \alpha + y \sin \alpha - s) dx dy \quad (1)$$

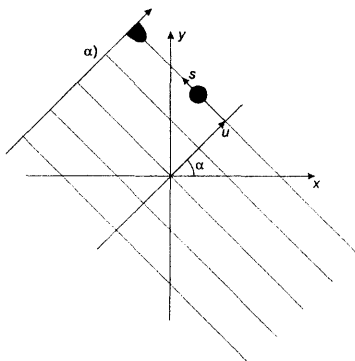


Figure 1. A single projection of a simplified object.

Projections performed in this way may be subjected to the next phase of signal processing: the values of the projection function $p(s, \alpha)$ passing through a fixed point (x, y) are accumulated to generate information about affiliation $\mu(x, y)$ at this point. The accumulation, as mentioned before and given by the following expression, is called the back-projection or the Radon's back-projection operator:

$$\tilde{\mu}(x, y) = \int_0^\pi p(x \cos \alpha + y \sin \alpha, \alpha) d\alpha \quad (2)$$

where:

$\tilde{\mu}(x, y)$ — the image of distribution of the x-ray affiliation in a section of the investigated body formed on the basis of the performed projections.

It is easy to show that the function $\tilde{\mu}(x, y)$ gives the image of the original distribution of affiliation $\mu(x, y)$ in the investigated cross-section of the object. The relationship between the function and the affiliation is given by [3]:

$$\tilde{\mu}(x, y) = \int_{-\infty-\infty}^{\infty \infty} \int_{-\infty-\infty}^{\infty \infty} \mu(x', y') \frac{1}{\sqrt{(x-x')^2 + (y-y')^2}} dx' dy' \quad (3)$$

The linear equation (3) has its discrete equivalent of the form:

$$\tilde{\mu}(i, j) = \sum_{k=1}^K \sum_{l=1}^L h_{ijkl} \mu(k, l) \quad (4)$$

where:

h_{ijkl} — gives a discrete impulse response of the signal, accountable for the geometric distortion of the original image.

The existing methods of image reconstruction from projection apply different ways of filtering projections $p(s, \alpha)$ to avoid the effects of geometrical distortion term, shown in expression (3). However, scientists keep looking for faster algorithms to shorten the time required to perform a complete reconstruction and have an image ready for diagnostics, they are also trying to improve the quality of the reconstructed image. In the above

context, the usage of neural networks could give a new impulse to the investigations of image reconstruction from projection.

2. The neural algorithm of image reconstruction from projections

For many scientists an adoption of entropy criterion seems to be a very attractive tool for the whole domain designated as image reconstruction. Their publication [2][5] presents methods of distortion elimination from images, reconstruction of incomplete images, using approach with maximization of the entropy of the distorted images. The neural network presented below realizes image reconstruction from projection using the maximum entropy criterion. Optimization taking advantage of the maximum entropy criterion in order to reconstruct discrete image can be formulated as the constraints:

$$\begin{cases} \max_{\mathbf{F}} Ent(\mathbf{F}) \\ \tilde{\mathbf{F}} = \mathbf{H}\mathbf{F} \end{cases} \quad (5)$$

The problem expressed in equation (5) can be reformulated using a penalty method and takes the form:

$$\max_{\mathbf{F}} \left(Ent(\mathbf{F}) + \text{wsp} \sum_{i=1}^I \sum_{j=1}^J f(e_{ij}(\mathbf{F})) \right) \quad (6)$$

where:

$$e_{ij}(\mathbf{F}) = \sum_{k=1}^K \sum_{l=1}^L h_{ijkl} \mu(k, l) - \tilde{\mu}(i, j) \quad \text{--- } i\text{-th row of matrix } \mathbf{H}\mathbf{F} - \tilde{\mathbf{F}},$$

wsp — suitable large positive coefficient,

$f(\bullet)$ — penalty function.

After conditioning in the expression (6) it can start to construct a neural network realizing the deconvolution task of the equation (4). One can formulate an energy expression to do it by minimizing the value of this function using neural network. Energy expression could take the form:

$$E^t = -Ent(\mathbf{F}^t) + \text{wsp} \sum_{i=1}^I \sum_{j=1}^J f(e_{ij}(\mathbf{F}^t)) \quad (7)$$

In order to find a maximum of this function its derivation is defined in the following form:

$$\begin{aligned} \frac{dE^t}{dt} = & - \sum_{i=1}^I \sum_{j=1}^J \frac{\partial Ent(\mathbf{F}^t)}{\partial \mu^t(i, j)} \frac{\partial \mu^t(i, j)}{\partial t} + \\ & + \text{wsp} \sum_{k=1}^K \sum_{l=1}^L \sum_{i=1}^I \sum_{j=1}^J \frac{\partial f(e_{ijkl}(\mathbf{F}^t))}{\partial e_{ijkl}} \frac{\partial e_{ijkl}}{\partial \mu^t(i, j)} \frac{\partial \mu^t(i, j)}{\partial t} \end{aligned} \quad (8)$$

A structure of the neural network realizing so formulated task is depicted in Fig. 2. Following this structure Fig. 3 presents a chosen cross-section of neural network, what makes analysing of the algorithm details more clear.

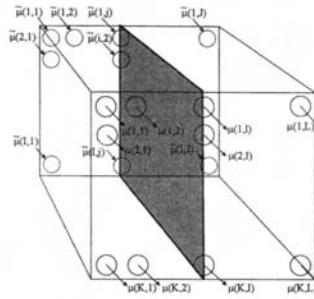


Figure 2. Structure of neural network

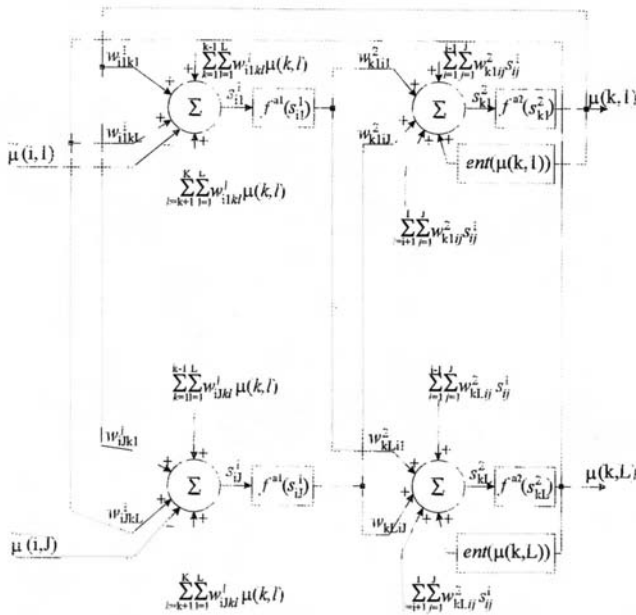


Figure 3. A cross-section of the neural network structure.

All symbols which appear in the neural network structure are listed according to the order from the input to the output of network:

I. layer

weights of connections: $w_{ij}^1 = h_{klj}$

weighting sum: $s_{ij}^1 = e_{ij}^1(\mathbf{F}) = \sum_{k=1}^K \sum_{l=1}^L w_{ij}^1(1) \mu^1(k, l) - \tilde{\mu}(i, j),$

activation function: $f^{a1}(s_{ij}^1) = \frac{\partial f(e_{ij}^1(\mathbf{F}))}{\partial e_{ij}^1}$

II. layer

weights of connections: $w_{ij}^2 = -C \frac{\partial e_{kl}^1}{\partial \mu^1_{ij}} = -C h_{klj}$

weighting sum:
$$s_{ij}^2 = \frac{\partial Ent(\mathbf{F}^t)}{\partial \mu^t(i, j)} + \sum_{i=1}^1 \sum_{j=1}^1 w_{ij}^2 f^{al}(s_{ij}^1),$$

where:
$$s_{ij}^2 = \frac{d\mu^t(i, j)}{dt},$$

activation function:
$$f^{a2}(s_{ij}^2) = \mu^t(i, j) = \int_0 s_{ij}^2 dt.$$

Additionally the following notation is used in the figure above:

$$ent(\mu^t(k, l)) = \frac{\partial Ent(\mathbf{F}^t)}{\partial \mu^t(k, l)}.$$

3. Experiments and their results

The above presented neural network structure was further tested using a mathematical model of the skull cross-section. During all simulations the image of this model with resolution 50×50 pixels was used. The progress in the reconstruction process is presented in Tab. 1. The first row in the table shows the number of realized iterations using the above described neural network algorithm of image reconstruction from projections. The second row consists of images arisen during the reconstruction process on the output of the neural network.






Number of iteration.	3	25	100	400	1000
View of the reconstructed image.					

Table 1. Process of image reconstruction from projection using a mathematical model of the cross-section of the skull.

One can see above that after about one thousand iterations the results of the reconstruction process are stabilized on a satisfactory level. Therefore, one can say that at this moment the image is reconstructed and the process can be stopped.

4. Conclusion

In the above shown context presented in this work, neural network could be a new impulse in the investigation of the image reconstruction from projection. Although the speed of the reconstruction process is still unsatisfactory, the perspectives of the parallel implementation of this neural network structure could accelerate calculations. Taking into account the fact that neural networks are one of the towers of soft-computing, their usage seems to be very tempting, in consideration of the many interesting profits from application of the artificial intelligence in this field of medical engineering.

References

- [1] Cormack A. M., Representation of a function by its line integrals with some radiological application, *J. Appl. Phys.*, vol. 34, pp. 2722-2727, 1963.
- [2] Frieden B. R., Zoltani C. R., Maximum bounded entropy: application to tomographic reconstruction, *Appl. Optics*, vol. 24, pp. 201-207, 1985.
- [3] Jain A. K., *Fundamentals of Digital Image Processing*, Prentice Hall, New Jersey, 1989.
- [4] Kaczmarz S., Angeneaherte Aufloesung von Systemen Linearer Gleichungen, *Bull. Acad. Polon. Sci. Lett. A.*, vol. 35, pp. 355-357, 1937.
- [5] Luo Fa-Long, Unbehauen R., *Applied Neural Networks for Signal Processing*, Cabridge University Press, 1998.
- [6] Radon J., Ueber die Bestimmung von Functionen durch ihre Integralwerte Tangs gewisser Mannigfaltigkeiten, *Berichte Saechsische Akad. Wissenschaften, Math. Phys. Klass.*, vol. 69, pp. 262-277, Leipzig, 1917.

New Approach to Estimation of Pulse Coupled Neural Network Parameters in Image Recognition Process

Radoslav FORGÁČ, Igor MOKRIŠ

Faculty of Finance, Matej Bel University Banská Bystrica, Slovakia

Abstract. The contribution deals with parameter estimation of Pulse Coupled Neural Network in feature generation part of image recognition process. The basic PCNN parameters, which directly influence the quality of generated features, are linking radius and PCNN kernel type. The contribution introduces a new approach to estimation of linking radius and its influence to image recognition process.

Keywords: *Pulse Coupled Neural Network, image recognition, feature generation, PCNN kernel, linking radius*

1. Introduction

The Pulse Coupled Neural Network (PCNN) is a biological model based on the mammalian visual cortex, proposed by Eckhorn [2]. The PCNN is advisable to solve tasks as the feature generation for image and pattern recognition [3, 4, 6] or image segmentation [9, 13]. Significant advantage of PCNN is the invariance to rotation, dilatation or translation of images [3, 6]. The structure of standard PCNN comes out from the structure of input image, which will be processed. It means that PCNN is single-layered, two-dimensional, laterally connected neural network of pulse coupled neurons, which are connected with image pixels each other. Several models of PCNN have been developed. The most used PCNN models are, for example, PCNN with a modified feeding input [1, 14], Fast linking PCNN [1, 7, 10] or Feedback PCNN [11]. More details of the PCNN functionality are described in [6, 7, 8, 12].

Each neuron of the PCNN has a specific structure (Fig. 1). It consists of an input part, linking part and pulse generator. The standard PCNN model is described as an iteration by the following equations:

$$F_{ij}(n) = S_{ij} + F_{ij}(n-1) \cdot e^{-\alpha_F} + V_F \cdot (M * Y(n-1))_{ij} \quad (1)$$

$$L_{ij}(n) = L_{ij}(n-1) \cdot e^{-\alpha_L} + V_L \cdot (W * Y(n-1))_{ij} \quad (2)$$

$$U_{ij}(n) = F_{ij}(n) \cdot (1 + \beta \cdot L_{ij}(n)) \quad (3)$$

$$\Theta_{ij}(n) = \Theta_{ij}(n-1) \cdot e^{-\alpha_\Theta} + V_\Theta \cdot Y_{ij}(n-1) \quad (4)$$

$$Y_{ij}(n) = \begin{cases} 1 & \text{if } U_{ij}(n) > \Theta_{ij}(n) \\ 0 & \text{otherwise} \end{cases} \quad (5)$$

where F_{ij} is the feeding input, L_{ij} is the linking input, n is an iteration step, S_{ij} is an intensity of pixel i,j in the input matrix, W and M are the weight matrices, $*$ is the convolution operator, Y is the output of the neuron, V_L and V_F are potentials, α_L and α_F are decayed constants, U_{ij} is the internal activity, β is the linking coefficient and Θ_{ij} is the dynamic threshold.

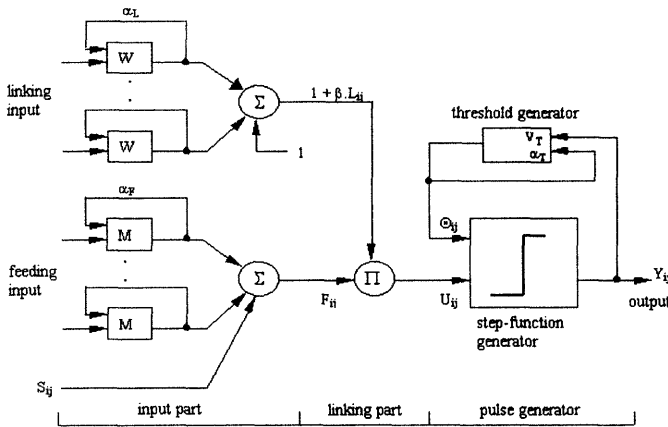


Figure 1. The standard PCNN neuron.

The convolutions $W*Y(n-1)$ and $M*Y(n-1)$ determine the local intensity of neurons, which are fired at the same time as neuron ij . The equations (6) and (7) subscribe these convolutions:

$$(W * Y(n-1))_{ij} = \sum w_{ijkl} Y_{kl}(n-1) \quad (6)$$

$$(M * Y(n-1))_{ij} = \sum m_{ijkl} Y_{kl}(n-1) \quad (7)$$

where the weights m_{ijkl} and w_{ijkl} are dependent on the distance between the neurons and on the selected kernel.

The time signal $G(n)$, generated from the sum of outputs Y_{ij} in every iteration step n is defined as:

$$G(n) = \sum_{ij} Y_{ij}(n) \quad (8)$$

A circle with radius r_0 , so-called linking radius, defines the linking area of every neuron in the PCNN. Only neurons in this linking area influence the neuron activity. The distance r between neurons in the linking area is defined as the Euclidean distance between the corresponding pixels of an input image.

Weights of connections between a neuron and its neighbors are commensurable to the distance between neurons. The values of weights are getting smaller with the increasing distance of neurons. Weights are dependent on the implemented PCNN kernel and on the linking area, because a linking radius is the basic parameter of the PCNN kernel distribution. The distributions like Gaussian, $1/r$ or $1/r^2$ are used very often as the kernel [1, 3, 4, 5, 9, 10, 13]. The calculation of weights are given by the following equations (9), (10) and (11) where couples ij and kl determine the position of two connected PCNN neurons:

$$w_{ij,kl} = \frac{1}{\sqrt{2\pi}} e^{-\frac{r^2}{2}} = \frac{1}{\sqrt{2\pi}} e^{-\frac{(i-k)^2 + (j-l)^2}{2}} \quad (9)$$

$$w_{ij,kl} = \frac{1}{r} = \frac{1}{\sqrt{(i-k)^2 + (j-l)^2}} \quad (10)$$

$$w_{ij,kl} = \frac{1}{r^2} = \frac{1}{(i-k)^2 + (j-l)^2} \quad (11)$$

The Gaussian kernel is used as a standard PCNN kernel for all other PCNN kernels, because it has the best response to input image [5]. On the other side, the computation of the Gaussian kernel is very time-consuming in comparison with simpler kernels. This fact is visible from equations (6), (7) and (9), (10), (11).

2. New approach to estimation of the linking radius

The goal of the contribution is optimization of the PCNN for a feature generation, to find the optimal compromise of the following activities: to maximize the speed of PCNN computing, to minimize the number of iterations, to reduce the number of features to an optimal value and together to obtain exact features for every tested input image. All activities must be solved at the same time. In this contribution we solve estimation of linking radius r_0 , that is the most important parameter for all activities listed above. Up to now the estimation of this parameter in many cases has been set experimentally [3, 4], therefore we introduce a new approach that automates the estimation process of an optimal linking radius. Others parameters, i.e. the linking coefficient β , decayed constants α_L , α_F , and α_T , potentials V_L , V_F and V_T , have the influence on PCNN output, but their optimization is not so important as the optimization of a linking radius. It is sufficient to set up these parameters in recommended intervals [5, 9, 13].

Our algorithm for automate estimation of a linking radius comes out from the comparing of the PCNN time signal with Gaussian kernel and the time signal with simpler kernel, e.g., $1/r$ or $1/r^2$. We expect the PCNN with any kernel would generate the same output time signal. However, simpler modifications of PCNN kernel have tendency to fragment the generated features by using lower values of the linking radius. We determinate the required linking radius r_0 by looking for the first local maximum of calculated totals of individual time signals. The same image raster and resolution are required for efficient implementing of this algorithm. The algorithm can be described in details as follows:

1. Define linking radius interval $(1, r_{max})$. Typically $r_0 \in (1, 4)$.
2. Bring the input image of tested set to PCNN input.
3. Set up tested PCNN, e.g. modified PCNN algorithm with $1/r$ or $1/r^2$ kernel.
4. Generate the set of features for $r_i=1$ to r_{max} , step $\Delta r=0.5$. Detailed partition of Δr has not practical effect for interval $r_0 \in (1, 4)$.
5. Calculate the total sum T_i of set of features in previous step for every r_i .
6. Find the first local maximum T_{max1} in the set of calculated totals.
7. If the local maximum is approximately equal to previous value, consider previous value to be local maximum. Repeat this step while distance between T_i values is minimal. So we get couples of total sum T_i and corresponding linking radius r_i .
8. Wanted linking radius r_0 for current input image responds to local maximum of calculated totals (see step 5).

9. Repeat steps 2 to 8 for every image in the tested set of images. This step is important if we want to estimate an optimal linking radius r_0 .
10. Sufficient linking radius r_0 for the tested image set is the most frequent value of chosen linking radiuses in step 8.

3. Experiments

We test and compare two types of PCNN kernels, namely Gaussian kernel and $1/r^2$ kernel. The standard version of PCNN has been used for Gaussian kernel and modified version of PCNN for $1/r^2$ kernel. Standard PCNN with Gaussian kernel is taken as reference to modified PCNN with $1/r^2$ kernel, because it has a very stable output for every linking radius. Our effort is to decrease duration of feature generation by comparable quality of generated features with the standard PCNN and Gaussian kernel. Therefore we are testing a modified version of PCNN which does not contain feeding input and computation algorithm is easier. On the other side, modified PCNN with $1/r^2$ kernel has worse results as mentioned PCNN with Gaussian kernel. So, we must find a suitable linking radius that solves our problem of quality of generated features.

Experiments described in this paper were realized on images of Slovak coins with nominal values of 1 Sk, 2 Sk, 5 Sk and 10 Sk. The image raster was 217×224 pixels, density 72 dpi and 24 bits represent one image pixel. The images of coins are relatively simple, but shapes are dull. The transformation of images of coins from the three-dimensional space to the two-dimensional space (e.g. by scanner or camera) more reduces these shapes. The other negative property is a monotony of coin color, where coins 1 Sk and 10 Sk are relatively color similar and coins 2 Sk and 5 Sk are color identical. Color negation of input images was used for better image extraction [5].

4. Results

Results of the experiments for image 1 Sk (face) are shown in Fig. 2, where the first local maximum for Gaussian kernel is given by $r_0 = 1.5$ and for $1/r^2$ kernel by $r_0 = 2$. These linking radiuses were sufficient for all tested images. The experiments have proved that increasing of a linking area does not have more significant effect on generated features when we use Gaussian kernel.

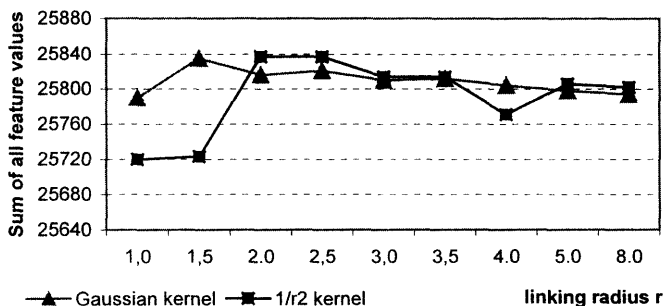


Figure 2. Total sum of a set of features for a specified linking radius r_0 . Results are shown for image – face of coin 1 Sk.

An increasing of a linking area above estimating value has a negative effect, because the response of the PCNN to input image rapidly grows. This feature is very undesirable in process of feature generation for image recognition.

We can claim that standard PCNN algorithm with Gaussian kernel gives us very good results with the linking radius $r_0=1$. When we use a simpler kernel and the linking area is minimized, i.e. $r_0=1$, the time signal with $r_0=1$ or $r_0=1.5$ is fragmented and in comparison with the results of PCNN with Gaussian kernel the time signals are not alike. On the other side, the time signals are alike when we use linking radius value estimated by our algorithm.

5. Conclusion

The goal of this contribution is an introduction of a new approach to estimation of linking radius and analysis of its influence on feature generation of image recognition process. We have used Gaussian PCNN kernel as standard, because it shows the best results from the tested kernels. Attention was given to decrease the duration of feature generation. Therefore the modified version of PCNN without a feeding input with $1/r$ and $1/r^2$ kernels was used. The experiments confirmed that our algorithm is useful for feature generation tasks. Our algorithm automates the estimation of sufficient value of the linking radius. Thanks to this algorithm we can save more time by finding of a suitable linking radius for the whole tested image set. So far we have been forced to find a suitable linking radius only experimentally.

References

- [1] Becanovic, V.: PCNN Toolbox for MATLAB ver. 5.1, 2001.
- [2] Eckhorn, R. et al.: Feature Linking via Synchronization among Distributed Assemblies: Simulations of Results from Cat Visual Cortex. *Neural Computation*, 1990, Vol. 2, pp. 293-307.
- [3] Forgáč, R., Mokriš, I.: Invariant Image Recognition Using Distributed Neural Networks. In: INES'99 - 3rd IEEE International Conference on Intelligent Engineering Systems, Stará Lesná, 1999, pp. 105-109.
- [4] Forgáč, R., Mokriš, I.: Invariant Representation of Images by Pulse Coupled Neural Network. In: E-ISCI'2000 - European Symposium on Computational Intelligence, Košice, Springer-Verlag, 2000, pp. 33-38.
- [5] Forgáč, R., Mokriš, I.: Parameter Influence of Pulse Coupled Neural Network for Image recognition. *Journal of Applied Computer Science*, Vol. 9, No. 2, 2001, pp 31-44.
- [6] Johnson, J. L.: Pulse Coupled Neural Nets: Translation, Rotation, Scale, Distortion and Intensity Signal Invariance for Images. *Applied Optics*, 1994, Vol. 33, pp. 6239-6253.
- [7] Johnson, J. L., Padgett, M. L.: PCNN Models and Applications. *IEEE Transactions on Neural Networks*, 1999, Vol. 10, No. 3, pp. 480-498.
- [8] Johnson, L. J., Ritter, D.: Observation of Periodic Waves in a Pulse-coupled Neural Network. *Optics Letters*, 1993, Vol. 18, No. 15, pp. 1253-1255.
- [9] Kuntimad, G., Ranganath, H. S.: Perfect Image Segmentation Using Pulse Coupled Neural Networks. *IEEE Transactions on Neural Networks*, 1999, Vol. 10, No. 3, pp. 591-598.
- [10] Kinser, J. M., Lindblad, T.: Implementation of Pulse-Coupled Neural Networks in the CNAPS Environment. *IEEE Trans. on Neural Networks*, 1999, Vol. 10, No. 3, pp. 584-590.
- [11] Kinser, J. M., Johnson, J. L.: Stabilized Input with a Feedback Pulse-Coupled Neural Network, submitted to *Optical Engineering*.
- [12] Kinser, J. M., Lindblad, T.: Inherent Features of Wavelets and Pulse Coupled Neural Networks. *IEEE Trans. on Neural Networks*, 1999, Vol. 10, No. 3, pp. 607-614.
- [13] Ranganath, H. S., Kuntimad, G.: Object Detection Using Pulse Coupled Neural Networks. *IEEE Transactions on Neural Networks*, 1999, Vol. 10, No. 3, pp. 615-621.
- [14] Ranganath, H. S., Kuntimad, G.: Iterative segmentation using Pulse Coupled Neural Networks. *SPIE*, Vol. 2760, pp. 543-554.

Prediction of Gross Domestic Product Development on the Basis of Frontal Neural Networks, Genetic and Eugenic Algorithms

Vladimír OLEJ

*Department of Information Systems, School of Finance, Matej Bel University
Tajovského 10, 974 01 Banská Bystrica, Slovak Republic
olej@financ.umb.sk*

Abstract. The paper presents the possibility of application of frontal neural networks, genetic and eugenic algorithms in predicting gross domestic product development by designing a prediction model whose accuracy is superior to the model used in practice [1]. The learning process is implemented by means of a newly designed algorithm based on the EuSANE algorithm [2].

Keywords: *Gross domestic product, economic variables, prediction, frontal neural networks, genetic and eugenic algorithms, EuSANE algorithm.*

1. Introduction

The most commonly used models in the field of economic variables (EV) prediction are models of time series extrapolation, single-equation regression models and multi-equation simultaneous models [3]. Neural networks (NN), genetic algorithms (GA) [4,5] and eugenic algorithms (EuA) [2,6,7,8,] provide in comparison with econometric methods of prediction the advantage of superior processing of non-linear dependencies [9] which may contribute to improvement of prediction accuracy [10].

Gross domestic product (GDP) is one of the most complex indicators of production activity in the economy. Gross domestic product represents the market value of the final goods produced by factors of production on the territory of the given state during the selected period. Even if business cycles are of different intensity and duration and are caused by various factors, there exist certain relations in the movement of EVs, which describe business cycles [11,12]. However, GDP remains the main indicator of production activity in the economy. Therefore, EVs used for the prediction of business cycles can be used for the prediction of GDP development.

2. Modelling of Gross Domestic Product Development

At present there are lots of EVs at disposal, which monitor the development of the economy, e.g. index of leading economic indicators (ILEI) and diffusive index of leading economic indicators (DILEI). The DILEI measures the percentage of decreasing and increasing EVs included in ILEI, i.e. the proportion of EVs indicating the reversal of economic growth. Both ILEI and DILEI can be used to model the prediction of GDP development. The interconnection of these two indicators provides a reliable signal of recession. It is assumed that recession starts if the average value of ILEI growth falls below 2 % and at the same time the value of DILEI drops below the critical value of 0.5. Until

now (2001) these indexes have not been used for the prediction of GDP development. The development of selected indexes of the US economy during the years 1965 - 1995 is shown in Fig. 1 and Fig. 2. The prediction accuracy can be described by means of Average Error ϵ , Mean Average Error σ and Root Mean Squared Error δ .

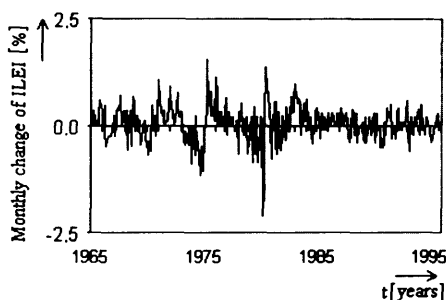


Figure 1. Development of ILEI of the US economy in the years 1965 – 1995

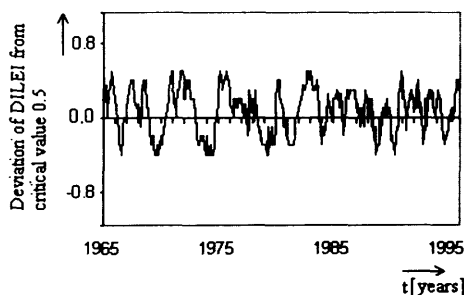


Figure 2. Development of DILEI of the US economy in the years 1965 - 1995

3. The Basic Notions of Neural Networks, Genetic and Eugenic Algorithms

The output signal of NN becomes a function of time when making the prediction. The time dimension can be implemented into NNs in two ways, namely implicitly and explicitly. Time becomes the proper output signal with regard to the explicit presentation of time. The implicit presentation of time represents the incorporation of the time dimension into the structure of NN. With regard to the implicit presentation of time, so-called filters [9], which apply short-term memory and operation of dimensional summation, can be used in feed-forward NNs. Dimensional summation represents the summing-up of input signals.

Short-term memory preserves the knowledge about states of NN surrounding, ultimately about individual neurons in the close past. There exist several possible structures of short-term memory from which the most commonly used is memory with a constant number of delays. This type of memory works with discrete time and with the operation of delay z^{-b} , where b is the number of delay time intervals. This operation works in the way that for the value of variable x in time t the value of variable x in time $t-b$ returns.

Every memory has its depth measured by parameter b . Deeper memory means that more information about past states of surrounding is preserved. As for feed-forward NNs, there exist two types of architecture with the implicit presentation of time, i.e. frontal and distributed NNs. In frontal NNs short-term memory is situated in the beginning of NN as a

fore-layer processing the time context. Classical back-propagation algorithms may train such NNs or another algorithm suitable for feed-forward NNs.

Genetic algorithms [4,5] are used as learning algorithms of NNs. Genetic algorithms differ in the quality of epistasis processing. Epistasis represents the dependency of genotype fitness η on the degree of interaction among alleles, eventually groups of alleles of genes. Epistasis occurs when a change of allele of a certain genotype or change of alleles of groups of genes causes an increase in the value of the objective function and at another time a decrease of this value, depending on alleles of other genes of the genotype. In this case it is not suitable to use GA, but the smart combination of alleles and groups of alleles is necessary. Eugenic algorithms enable the smart recombination dependent on the analysis of interaction among alleles and their groups [2,6,7,8]. Eugenic algorithms use principles of so-called eugenic evolution. Population develops according to the eugenic principles if genotypes with the smallest fitness η are gradually replaced by new genotypes generated on the basis of the smart recombination of genotypes of the entire population. An EuA can be, for example, expressed in the following way [2,6,7,8]:

```

1.  P = random-pop ( )           /* Randomly initialise genotype population */
    L = 0                         /* L is number of new individuals created by EuA */
2.  DO WHILE ( L < Lmax )        /* Create new individuals */
    G = { g1, g2, ..., gi }      /* G is set of empty genes */
    rpop = pop                     /* rpop is the population after the restriction */
    DO WHILE nonempty(G)          /* Select empty gene and assign allele to it */
    gg = the_most_significant_gene ( rpop, G ) /* gg is the most significant gene */
    gg,a = select_allele ( gg )   /* gg,a is allele assigned to gg */
    Remove gg from G             /* Restrict the population if E is high */
    E = epistasis_pop ( rpop, G ) /* E is population epistasis of rpop */
    pr = p_restriction ( E )      /* pr is the probability of the restriction */
    IF uniform-random [ 0,1 ] < pr THEN
    10. rpop = restriction_pop ( rpop, xg,a )
    END                           /* Replace U by newly generated genotype */
    11. U = the_most_unfitting_genotype ( pop ) /* U is the most unfitting genotype of pop */
    12. popa = X, L:=L+1          /* X supplies U in the population */
    END

```

4. The Model of Gross Domestic Product Prediction

The model of GDP prediction is designed for the unconditional prediction of real GDP development of the USA (which is expressed as percentage change of value of produced GDP against last quarter) whereas the moment of the prediction is 6 months before the beginning of the prediction period. The model of the prediction is presented by NN with the use of the filters applying short-term memory with a constant number of delays. The formulation of frontal NN with the same memory depth of all filters, linear filter and linear output neuron is as follows

$$Y = \sum_{k=1}^K \alpha_k d \left(\sum_{j=1}^J \beta_{jk} \sum_{i=1}^b \chi_{ijk} X_{ijk} \right), \quad (1)$$

where Y is the output of NN, α is synapse weight vector among neurons in the hidden layer and output neuron, β is synapse weight vector among filters and neurons in the hidden layers, χ is synapse weight vector inside the filter, k is index of neuron in the hidden layer, K is the number of neurons in the hidden layer, d is activation function, j is index of the filter, J is the number of filters per neuron in the hidden layer, b is short-term memory depth of the filter, X is the output vector of NN.

The number of filters in the input layer of frontal NN is dependent on the number of EVs used in the model and their time delay. The algorithm selects the inputs of frontal NN. It would be necessary to allow the algorithm to choose the length of the input vector. Thus the number of input neurons becomes the parameter of the model. The model uses the linear

combination of the inputs, because time series of EVs consist of significant linear elements. Therefore the activation function should be sufficient to cover the non-linearities. Neurons used in the hidden layers consist of frequently applied logistic functions with slope parameter equal to 1. The output neuron is linear.

5. The Learning Process and Analysis of the Results

The learning algorithm looks for such combinations of EVs, time delays and weights which lead to the least prediction error as regards tendency and size of GDP change. Thus the algorithm realises not only the prediction of GDP, but also the selection of suitable EVs, including time delays. Therefore, it is necessary to design the learning algorithm which enables to leave local minima and to continue to search in various segments of the space of all possible solutions.

One of the suitable alternatives is the application of a GA. A GA can be considered as a faster algorithm than an EuA, because a GA does not compute epistasis. However, in this case epistasis analysis appears to be useful because of dependency of statistical significance of the additional EV of the model on the EV already included in the model, therefore application of the EuA would be justified too.

The Eugenic Symbiotic Adaptive Neuro – Evolution (EuSANE) algorithm, which allows taking advantage of both algorithms, is presented in [2,6,7,8]. A new learning algorithm has been proposed for the model of frontal NN on the basis of advantages of the EuSANE algorithm. Except for its presentation of neurons it differs from [2,6,7,8] in that it does not work in two but in three grades. In the first grade the population of P0 filters associated with the input neurons is developed by means of a GA. In the second grade the population of P1 neurons in the hidden layer is also developed by means of a GA. In the third grade the population of P2 hidden layers is developed by means of an EuA. The population initialisation of the proposed EuSANE algorithm is random. The end criterion of algorithm is the maximal number of generations.

Prediction quality should have two dimensions: deviation size of predicted value of GDP growth from real value and number of errors when predicting the tendency of GDP change. These criteria influence the construction of the objective function h . A suitable form is as follows

$$h = \frac{1}{1 + O a_1} \frac{1}{1 + \delta a_2}, \quad (2)$$

where O is the number of errors when predicting the tendency of GDP change, δ is the root mean squared error of the prediction and a is the parameter of the objective function. This function is maximised by means of the newly-designed EuSANE algorithm to find the suitable economic hypothesis and the suitable parameter values of the prediction model. The aim goal of the algorithm is to find the combination of the independent variables and parameters of the model to obtain higher accuracy of the prediction in comparison with the FRBSF model [1]. Gross domestic product prediction is shown in Fig. 3. Table 1 consists of the average error values¹ of the prediction for the years of the learning set, the testing set and for all years of prediction

¹ Mean growth rates were obtained as yearly geometric averages of quarterly growth rates.

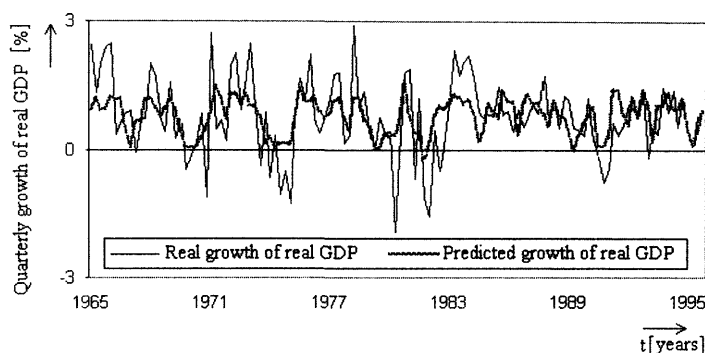


Figure 3. Prediction of quarterly growth of real GDP by means of frontal NN

Indicator of the prediction error / Determination of time interval for which average error of the prediction is calculated	All years (1965-95)	Learning set (1965-80)	Testing set (1980-95)	Years allowing comparison with FRBSF model (1985-95)	Reference values of FRBSF model (1985-95)
E	0.0679	- 0.4313	0.4307	0.4555	approximately zero
Σ	1.2815	1.1816	1.3751	0.8088	1.8451
Δ	1.7738	1.6099	1.9148	1.1298	2.5111

Table 1. Values of the average error indicators of frontal NN prediction

It can be concluded on the basis of the data in Table 1 that a mean prediction error of approximately zero indicates the absence of systematic errors in the model. The value of indicator δ reveals the existence of several larger individual errors of the prediction. The proposed model achieves, within the indicator of mean errors, superior accuracy (with reference to the FRBSF model). Twice as good accuracy was achieved by the designed frontal NN when taking into account the δ criterion, which puts emphasis on large individual deviations from the prediction. From the analysis it results that EVs and parameters of the model were found, which within given criteria provided superior accuracy of the prediction with reference to the FRBSF model.

6. Conclusion

The aim of the paper has been to show the possibilities of utilisation of frontal NNs, GAs and EuAs in the prediction of GDP development by designing a prediction model, which provides superior accuracy to the existing reference model. A model of GDP development formulated as frontal NN was designed to fulfil this goal, whereas parameter estimates and selection of EVs were realised by the newly-designed three-grade EuSANE algorithm. Frontal NN uses the ILEI and DILEI as EVs of GDP development. This model was compared with the FRBSF prediction model used in practice while mean error indicators of prediction were used as quality criteria. The model was designed by a combination of frontal NN, GA and EuA, which appeared to be more accurate within given criteria. On the other hand, the existence of several high individual errors limits the reliability of the proposed model. The existence of high individual prediction errors can be reduced by

incorporating the given criteria into the objective function of the algorithm of the parameter estimates of the model.

References

- [1] Ingenito, R., Bharat, T. (1996) Using Monthly Data to Predict Quarterly Output. FRBSF, Economic Review, No. 3.
- [2] Polani, D., Miikkulainen, R. (1999) Fast Reinforcement Learning through Eugenic Neuro - Evolution. Technical Report AI 99-277, Artificial Intelligence Laboratory, The University of Texas at Austin.
- [3] Pindyck, R. S., Rubinfeld, D. L. (1998) *Econometric Models and Economic Forecasts*. McGraw-Hill, Inc.
- [4] Holland, J. H. (1975) *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.
- [5] Olej, V. (1999) Evolution Stochastic Optimization Algorithms: Genetic Algorithms and Evolution Strategies. *Journal of Electrical Engineering*, ISSN 1335-3632, Vol. 50, No. 9-10, 266-272.
- [6] Prior, J. W. (1998) Eugenic Evolution for Combinatorial Optimization. Report AI 98-268, Artificial Intelligence Laboratory, The University of Texas at Austin.
- [7] Moriarty, D. E., Miikkulainen, R. (1998) Hierarchical Evolution of Neural Networks. *Proc. of the IEEE Conference on Evolutionary Computation, ICEC – 98*, Anchorage.
- [8] Moriarty, D. E., Miikkulainen, R. (1996) Efficient Reinforcement Learning through Symbiotic Evolution. *Machine Learning*, Vol. 22, 11-33.
- [9] Haykin, S. (1999) *Neural Networks*. Prentice-Hall, Inc.
- [10] Tkacz, G., Hu, S. (1999) Forecasting GDP Growth Using Artificial Neural Networks. Department of Monetary and Financial Analysis, Bank of Canada, Working Paper 99-3, ISSN 1192-5434.
- [11] Burda, M., Wzplosz, CH. (1993) *Macroéconomie, une perspective européenne*. De Boeck Université.
- [12] Kettell, B. (1999) *What Drives Financial Markets?* Financial Times – Prentice Hall.

A Fuzzy Logic Approach for the Condition Monitoring of Rotating Machines

Jens STRACKELJAN

Technical University of Clausthal, Institute of Applied Mechanics
38678 Clausthal-Zellerfeld, Graupenstr. 3, Germany
jens.strackeljan@tu-clausthal.de

Abstract. This paper examines the use and benefit of fuzzy technologies for solving the problem of machine condition monitoring. As a solution to this, the diagnostic system described here involves the measurement of machine vibration to detect faults. Some experimental results are given comparing the classification results obtained for higher derivatives with acceleration signals. In these applications, the quality of the input features for a fuzzy pattern classifier is significantly higher if features are calculated on the basis of a higher derivative of the time signal than on the basis of the original measured acceleration signal.

1. Introduction

The use of advanced pattern recognition systems in assuming an objective perspective in statements concerning the state of technical systems has gained increasing importance. Thus the economy of highly automated and cost intensive machines can be guaranteed only upon the high availability of these machines. The use of advanced, high-performance monitoring and diagnosis systems can make a significant contribution to this. Certain processes can be carried out safely for man and the environment only by means of reliably operating machines, particularly in fields where safety and environmental aspects play an important role.

2. Expert systems based on fuzzy technologies

In the automatic control of technical systems, supervisory functions serve to indicate undesired or non-permitted machine or process states and to take appropriate actions in order to maintain operation and to avoid damage or accident. The following basic functions can be distinguished:

1. Monitoring: measurable variables are checked with regard to tolerances, and alarms are generated for the operator.
2. Automatic protection: in the case of a dangerous process state, the monitoring system automatically initiates an appropriate counteraction.
3. Monitoring with fault diagnosis: based on measured variables, features are determined and a fault diagnosis is performed; in advanced systems, decisions are made for counteractions.

The advantage of the classical level-based monitoring (1 and 2) is simplicity, but these are only capable of reacting to a relatively large-scale change in a given feature. If the early detection of small faults, and a fault diagnosis, are desired, advanced expert systems based on the Fuzzy Technology or Neural Networks could be used. The basic element in an

expert system is the knowledge base which consists of expert information on the problem area in the form of facts and rules that are then combined to make conclusions by means of an inference machine. Such expert systems are said to simulate the conclusions made by a human expert in the case of relationships difficult to define using algorithms. For this purpose, it is necessary to record as much information as possible as rules in the form:

if premise is fulfilled **then** conclusion is valid

Fig. 1 shows, with reference to the detection of cracks in a rotating shaft, a typical rule which can be regarded as state of the art. Curves represent the relative values obtained by means of the first (f), second ($2\cdot f$) and third ($3\cdot f$) running speed components and the RMS value when the measurements are performed using shaft displacement.

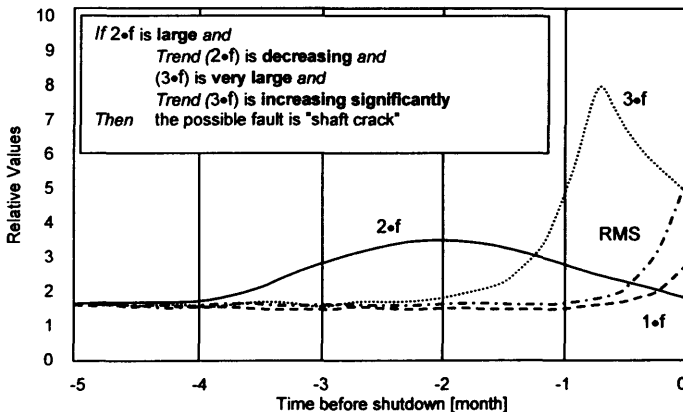


Figure 1. Trends in some characteristic values for the detection of cracks in rotating shafts

Automatic decision-making regarding the current machine status then takes place on the basis of these rules where, as opposed to the use of Neural Networks, the explanation component makes allows for making each system-related decision with reference to the knowledge base. The relationship between the structure and the amplitudes of a vibration signal and the cause of the defect in the technical object can scarcely be described completely and sufficiently in the form of rules. Classification procedures that arrive at a decision on the basis of pre-assigned, thus learned examples are frequently superior here. In practice, knowledge involves uncertainty factors that only allow for approximating inferences.

One method that accounts for knowledge uncertainties is the fuzzy logic developed by Zadeh in the mid-sixties. In contrast to the classical logic, which only allows two values to be either true or false, or 1/0, the fuzzy logic involves multiple values and allows all values between 0 and 1. This means that a set of values, called fuzzy sets, can also receive non-integer values. Relative to a crisp classification system, an object is no longer assigned unambiguously to one class only, but scalar class membership functions are employed instead to obtain membership values that will serve as a basis for deciding on class membership. In the case of multiple damage, crisp classification systems frequently involve considerable misclassifications, which in wearing processes that do not show a clear separation between the various fault classes will lead to a loss of information in the transition zone. For this reason, the classifier derived in the present work uses a classification algorithm based on fuzzy data.

3. Monitoring the condition of rotating machines

Safety-relevant or production-relevant plant components are already subject to monitoring at present, and the measurement of vibration signals is a suitable diagnosis parameter. However, most of the commercially available machine monitoring systems are not sufficiently well suited for performing diagnostic functions. The following shortcomings in particular are worth mentioning:

- insufficient stability toward noise load and interference (cavitation)
- utilization of only one, or very few diagnostic parameters
- susceptibility to severe fluctuations in measured signals
- limit-value-oriented monitoring of a restricted frequency ranges
- an inadequate method for classifying specific industrial process malfunctions.

From the point of view of its condition, there could be a significant change in the vibration behavior of a pump, for example, but the high background noise level makes it impossible to detect these changes in a signal by merely measuring the overall level or some other obvious characteristics.

A decided improvement in diagnostic reliability can be achieved by the simultaneous consideration of several vibration features and the supplementary processing of as many process parameters as possible, such as reactor pressure, temperature, power etc. In this conjunction, no competitive alternative is currently available to the application of classifying pattern recognition methods to the automatic appraisal of a technical object. As a result, a multi-dimensional feature vector which includes all measured data as components must be projected onto a scalar value to describe a class membership. This parameter is then employed for deciding whether the actual feature vector is included in one of the previously defined damage classes, and in which of these. This task can be performed continuously and fully automatically by a computer, without having to consult a human expert to conduct the diagnosis. In machine monitoring, it is evident that a model capable of describing the relationship between the cause of damage and the vibration signal mathematically is only seldom available for such complex combinations of many machine components under the influence of external fluctuation parameters. The efficiency of the associated classification algorithm is thus determined essentially by a 'learning process' that precedes the diagnostic process itself. In a learning process of this kind, certain free classifier parameters must be specified on the basis of unambiguous, pre-classified representatives of the various classes. During the subsequent working phase, unknown objects which 'resemble' the previously recorded learning example only to an extent yet to be specified are then independently assigned to one of the previously defined significance classes. An essential aid for designing a classifier of this kind is the mathematical formulation of fuzzy sets.

As a practical application of the diagnostic system, we will describe here the early detection of defects in roller bearings. The initial stage of bearing damage is characterized by the release of very small material particles from the running surface, by the appearance of fatigue cracks near the surface, or by the rupture or bursting of bearing components. The geometrical alteration of the race is revealed by the generation of a short shock when the roller passes over the damaged point. The shock propagates in the form of an approximately spherical wave to the front of the site of origin and excites nearly all bearing components capable of vibration to characteristic vibrations in the corresponding frequency range. To detect bearing faults, we calculated 24 secondary features on the basis of the vibration signal by means of a computer-aided data acquisition system. The implemented classifier calculates the membership values and provides a complete appraisal of the current state of the bearing at intervals of 1 min. For the convenience of the operating personnel, the

membership values are only determined for the four global classes, “shut-down”, “intact state”, “change in condition” and “fault”. The shut-down class describes a situation status where the machine is not in operation. The advantage of projecting 24 characteristic values onto a scalar value, which can be followed without difficulty on a trend plot, is evident. If the feature vector is considered more exactly, of course, the system can provide information that extends beyond the pure representation of class membership. The operator receives the classification results in a representation, which provides a quick overview of a couple of monitored machines.

Fig. 2 illustrates the learning set for the four defined classes for the sole consideration of the Kurtosis features and the KPF value. Kurtosis, which refers to the fourth order moment about the mean normalized by σ^4 , where σ is standard deviation, is used to measure the peakiness of a signal. KPF is a characteristic damage frequency that can be calculated from the bearing geometry and shaft rotational frequency. A bearing fault can be detected from the frequency spectrum at the KPF, i.e. by means of the KPF value. Index 1 in Fig. 2 indicates that the corresponding frequency is calculated for the inner ring. Only very slight scattering of the feature vectors occurs for the exactly defined shut-down class, whereas the scattering increases considerably for the “change” and “defect” classes.

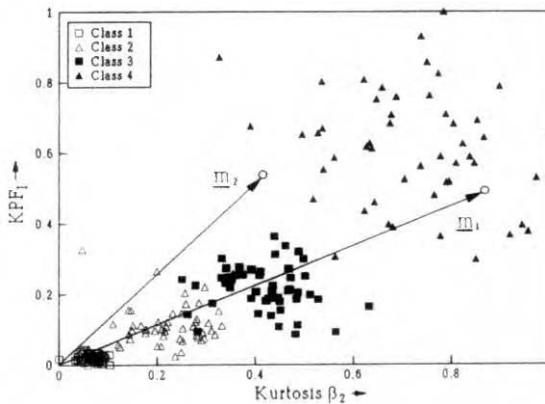


Figure 2. Learning samples for the diagnosis of roller bearings

An example of the importance of recording more than one process parameter during the diagnosis process is presented in Fig. 3. It shows a typical time signal for a roller bearing with outer race damage (left) and the signal for the same roller bearing and defect with the exception that its nature is different from the result of cavitation (right). In this case the kurtosis value decreases to the value for a nearly intact bearing because the cavitation reduces peakiness and would now not allow for drawing any conclusions about machine condition. In combination with the KPF values, however, the diagnostic system provides a correct appraisal that is largely independent of external effects.



Figure 3. Time record showing a typical fault signature of a defect roller bearing without (left) and with cavitation interference (right)

The example presented here indicates the suitability of the fuzzy pattern recognition approach, particularly for designing a monitoring system for rotating components for which an analytical description of the relation between cause and effect cannot be provided. At a primary chemical industry plant, the diagnostic system described here has been in use for more than four years to monitor several continuously operating centrifuges, and its effectiveness has already been proved repeatedly by the early detection of 12 roller bearing faults [Str99a].

4. Fuzzy classification of higher derivatives of the time signal

The traditional measurement parameters used in condition monitoring are displacement, velocity and acceleration. It was shown in some publications that fault detection can be improved if the classifier uses higher derivatives of the time signal [Lah95,Lah97,Str99b]. As an example of the suitability of higher derivatives as a data pre-processing technique to generate input features for a pattern recognition system, we will here present results obtained from detecting faults in roller bearings. The investigations focused on very slight damage so that failures were induced by making small grooves in the outer race and changing the speed in the range 1 - 40 Hz. Class 1 represents a bearing without damage and class 4 a groove of a width 1.0 mm, while classes 2 and 3 represent slight and medium damage with a groove between 0 – 1.0 mm.

A practicable way of implementing is to test the efficiency of the algorithm so as to classify measurements into the four specified damage classes with the smallest possible classification error. The results of the classification based on the classification algorithm are listed in the table for the various features that are used for classifying input information. It

Features	Acceleration		Derivative $x^{(3)}$	
	Error	$\Delta \mu$	Error	$\Delta \mu$
2	17 %	0.26	4 %	0.50
3	15 %	0.24	2 %	0.42
4	12 %	0.22	1 %	0.37
5	11 %	0.19	1 %	0.39

is evident that the use of two features only does not guarantee an error-free reclassification of the learning set. The reclassification error shown in the table is based on a test set of 160 random samples. The test also contains the average difference $\Delta \mu$ of the calculated class

membership values. The classification results are much better if features are considered which are calculated on the basis of the derivative $x^{(3)}$ than by means of the acceleration signal [Lah99]. For the task of the automatic feature selection the software FeatureSelector [Str99b] was used. The representation (Fig. 4) of the features crest and kurtosis in the feature space evidently indicate that the derivative $x^{(3)}$ generates more compact clusters for the distribution of the 40 learning samples for each class than does acceleration. The rate of misclassification using crest and kurtosis was 17 % for the acceleration and 9 % for the higher derivative.

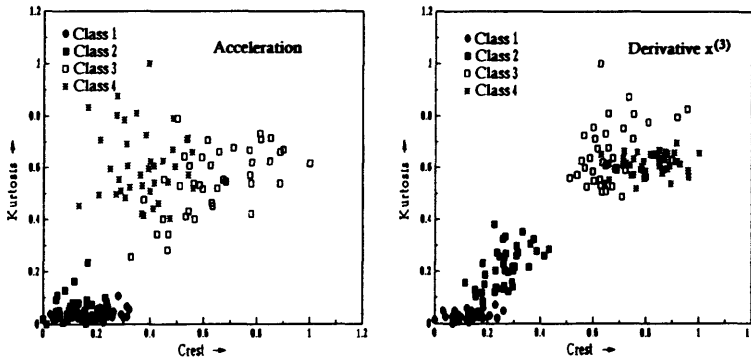


Figure 4. Distribution of crest and kurtosis features using acceleration (left) and the higher derivative $x^{(3)}$ of the time signal (right).

References

- [1] Lahdelma, S., (1995): "On the higher order derivatives in the laws of motion and their application to an active force generator and to condition monitoring," Academic Dissertation, Research report No. 101, University of Oulu.
- [2] Lahdelma, S. (1997): On the Derivative of Real Number Order and its Application to Condition Monitoring. *Kunnossapito* 11 (1997) 4, pp. 25-28.
- [3] Lahdelma, S., Strackeljan, J., and Behr, D., (1999): Combination of Higher Order Derivatives and a Fuzzy Classifier as a New Approach for Monitoring Rotating Machinery, *Proc. of COMADEM 99*, University of Sunderland, UK, pp. 231-241.
- [4] Strackeljan, J., and Behr, D., (1996): Condition monitoring of rotating machines using a fuzzy pattern recognition algorithm," In: *Vibration in rotating machines*, Oxford, pp. 507-512.
- [5] Strackeljan, J., and Weber, R., (1999a): Quality Control and Maintenance, In: *Fuzzy Handbook: Practical Applications of Fuzzy Technologies*, Vol. 7, Kluwer Acad. Publishers.
- [6] Strackeljan, J. (1999b): Feature Selection Methods for Soft Computing Classification. In: *Proceedings of the ESIT 1999 -European Symposium on Intelligent Techniques*, Kreta.
- [7] Strackeljan, J., Lahdelma S. a. Vuoto V., (1999c): Vibration monitoring of slowly rotating bearings using higher derivatives and a fuzzy classifier. In: *Condition Monitoring 99*, University of Wales Swansea, UK.
- [8] Vyas, N.S.; Satishkumar D. (2001): Artificial neural network design for fault identification in a rotor-bearing system. In: *Mechanism and machine theory*, 36: 157-176.
- [9] Zimmermann, H.-J. (1991): *Fuzzy-Set-Theory and its applications*. Kluwer Academic Publishers, Boston.

Neural Networks for SISD Real Time Control

Čapkovič Miroslav and Kozák Štefan

Department of Automatic Control Systems

Faculty of Electrical Engineering

Slovak University of Technology

Bratislava, Slovakia

fax: +421-2-65249734 <http://www.kasr.elf.stuba.sk/>

e-mail: miro@kasr.elf.stuba.sk e-mail: kozak@kasr.elf.stuba.sk

Abstract. Due to complexity of neural networks the following problems appear: (1) extensive requirements on computational devices where the network is trained or evaluated, (2) slow network evaluation/training response. The problem is more evident when using networks on devices where conventional algorithms run without problems with respect to the time response. In general this problem has to be solved using a more powerful computational device or parallel systems.

This paper deals with a neural network optimized for running on a single (SISD) end-device to obtain real time response in real time control systems and in this paper special effort is made to introducing the associated problems and designed solutions.

Keywords: Neural networks, real time, function approximation, network structure optimization

1 Introduction

Recently the phenomenon of neural network has been penetrating into various scopes of life. Although the neural network was first mentioned in the seventies, real applications appeared only after introducing the back-propagation algorithm. At present, neural networks are used to solve different computational tasks ranging from relatively simple identification methods up to complex tasks of artificial intelligence. Of course, many network structures and learning algorithms have been explored after introducing the back-propagation algorithm, but due to its good approximation precision and capability to solve the problems described by (input-output) data relations it is still applicable and sufficient to solve control problems without exact knowledge of mathematical model.

1.1 Motivation

As it is well-known the functionality of neural networks is based on a large number of cells interconnected into a network structure. Such structures bring the power of massive parallelism into the network algorithm computation and thus a network is represented by a number of concurrently working computational units, however on the other hand, this network features a large computational effort required for network computation algorithms (learning, evaluating). Using a single computational unit (end-device) brings about long computation time which does not permit neural networks to be run on standard computational devices (microprocessors at lowest clock frequency) frequently used for conventional algorithms to obtain real time responses. This problem also occurs when the network algorithm is performed on a large data set.

1.2 Problem description and identification

Our task is to optimize NN algorithms response time **without** affecting its functionality, which partially covers approximation and generalization abilities of networks optimized in this manner. Because at present the algorithms are *tightly depend on computers* or some computation-able devices on which they run, the optimization of transformation from algorithm symbolic representation into executable code should be taken into consideration (this dependence has a greater effect when the NN algorithm is in interaction with the real time system).

These optimization techniques should cover optimization on the highest level (the level of algorithm, its task, steps performed by algorithm, ...), on the medium level (algorithm's representation in the machine-human interface, which generally is a higher programming language) and on the lowest level (optimal translation of such optimized code into end device executable). According to our identification, major problems in this transformation with respect to the neural networks are:

- **High data complexity and redundancy.** The problem is related to the data redundancy in training set, data relevancy, large data scale, complex input-output relations and large training sets.
- **Network structure redundancy.** In general, the problem is involved in the redundant network structure which brings about multiple computational overhead. The number of nodes, number of layers and number of nodes edges should be taken into consideration with respect to the network redundancy together with used elementary activation functions.
- **Network activation function complexity.** The high computational complexity is a result of highly time and computation expensive elementary functions used as activation functions. As it is shown later, the majority of network computational time is spent by the elementary (activation) function computation.
- **Algorithm efficiency.** The problem covers the non efficient logical constructions in the algorithms and also non efficient higher language constructions.
- **Compiler and device overhead.** Problems are due to overheads brought about by higher programming languages and dynamic architecture of the NN representation, especially:
 - Overheads produced by parameter copying and stack operations (if any) which are:
 1. program function parameter passing
 2. program function call
 - Overhead due to the access to internal variables and structured data representing network parameters.
 - Overhead due to the access to learning set (off-line learning).
 - Overheads produced by accessing variables representing the solved problem.

We define the term *data access overhead* as overhead produced by programming techniques on accessing data structures. Frequently, it arises when higher (structural) languages are used for network representation and when the access to a network parameter generates N access steps instead one (problem of pointer arithmetics). The term *function and code call overhead* is associated with the overhead which is produced by calling pre-compiled libraries or defined functions. The overhead code is produced by parameters copying and the method how this function access to its local variables (reference by address or value).

	Original $\exp()$	Poly. approx.	series reduction N=8
Precision	-	$> 1e + 6$	$> 1e + 6$
Interval	-	$< -10, 10 >$	$< -10, 10 >$
Additions	-	4	8
Multiplications	-	4	8
Other	-	0	0
Time[s] ¹ .	3.22	1.01	2.04

Table 1: Computation complexity of $\exp()$ function replacement algorithms.

2 Optimization levels for solving specified problems

As already mentioned above, the major task of our paper is to prepare the network for usage in real-time systems. To solve this problem we define performance steps which are introduced below and explicitly explained in [1]. This paper shortly explains the optimization levels designed for identified problems. As can be shown, the introduced optimization techniques can be used separately, but order of optimization levels should be preserved. We present six optimization levels for optimizing networks representation in computational devices (processors, microcomputers and others). The following optimization levels were designed for particular problems:

2.1 Data preprocessing and postprocessing.

Optimization level preprocesses data by (1) normalization technique algorithms to obtain the original value compression within specified interval and (2) algorithms for redundant data elimination or (3) data transformation (linear or also nonlinear). Generally this optimization level can help to optimize the final network structure and its algorithms.

2.2 Local observers strategy.

The philosophy of *local observer* generally acts as a factor which decompose the original problem (whole state space) into simpler problems (local spaces). This strategy generally leads to simplification of local computation problem. We adapt this philosophy with respect to the computation time consumption optimization of computation process, when complex original network is replaced by simpler NN local observers performing.

2.3 Network structure optimization.

This level solves the problem of **network optimal structure** using the *structure optimization* algorithm introduced in [1]. This algorithm covers the network optimal structure design and node activation functions design with respect to the network qualitative indicators and **searches** better (till to optimal) architecture of network with respect to the fitness function and obtained performance indicators (generalization ability and approximation ability $\Delta(\Pi, \theta) = (\mu_G(\Pi, \theta), \mu_A(\Pi, \theta))$) for problem Π .

2.4 Elementary function reduction.

In this optimization level the original node elementary activation function is replaced by a less computationally expensive function (approximated function, spline approximation or another less computationally expensive function) with approximately same properties.

	Original $\exp()$	Simpl. poly. approx.	Ext. poly. approx.
Precision	-	1e+6	-
Interval	-	$< -10, 10 >$	$< -10, 10 >$
Additions	-	4	N_{\max}^2
Multiplications	-	4	N_{\max}
Other	-	1 division + 1 lookup	$3\log_2(M)^3$
Time[s] ⁴ .	3.22	1.03	-

Table 2: Computation complexity of $\exp()$ function replacement algorithms.

code	cycles	execution time [s]
function with overhead	10000000	0.69
C inline function	10000000	0.66
reduced function	10000000	0.17

Table 3: Comparison between function optimized by call reduction and original function. Tests were performed using the tool "time" on Linux box. The time covers only user processor time.

2.5 Device and algorithm optimization.

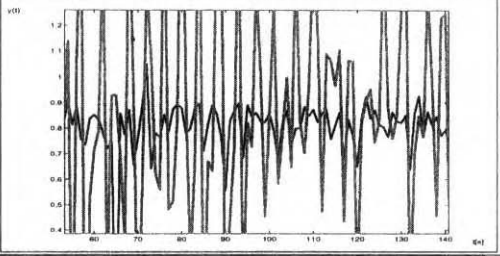
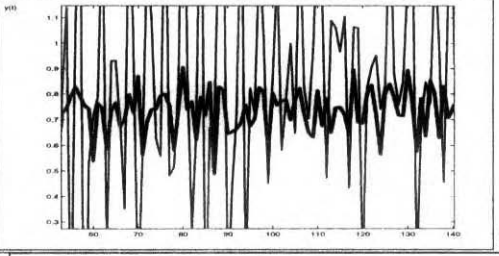
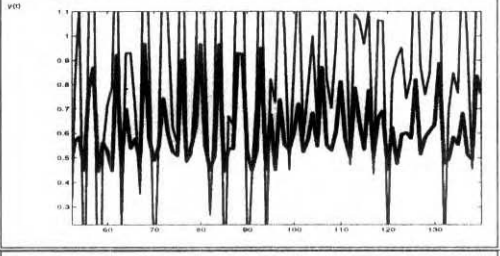
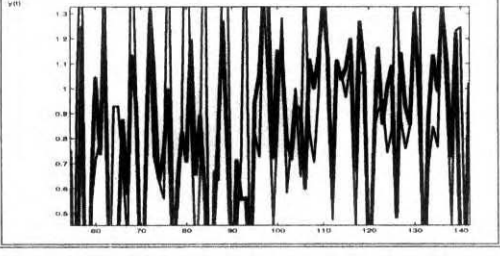
This optimization level covers the reduction of **device and algorithm** associated problems. The problem of data access overhead is reduced using the technique of replacing the N-nary step access by the 1-nary one for variables, and 0-nary for constants (compiled-in data). The problem of code access overhead is reduced using a technique similar to the function in-lining technique. This optimization level generates linear code containing only useful code (generally only math instructions.).

2.6 Linear code optimization.

This optimization level is applied to the generated linear code ξ , if there still persist some overheads in the linear code which are produced by the dependencies in the algorithms and the network structure. The algorithm of linear code optimization and data dependency reduction is applied on the generated linear code where it reduces the multiple occurrence of the same chains of instructions operating on the same data. After generating an overhead free linear code, we can translate the device-independent code into the specific device natural language.

code type	network evaluating time (ix586) [s]
original code	5.6
data access reduction to 1-NARY	3.48
data access reduction to 0-NARY	3.43

Table 4: Comparison between standard network programmed in a higher language and a network with data access reduction.

Replacement	MSE	Figure
poly. ()	247.835	
Series N=2.	0.0438438	
Series N=4.	0.0415324	
Series N=8.	0.0260172	

5

Table 5: Neural network MLP architecture 15,1. Comparison of networks with original activation function and the replacing function for *exp()* function.(Output time responses)

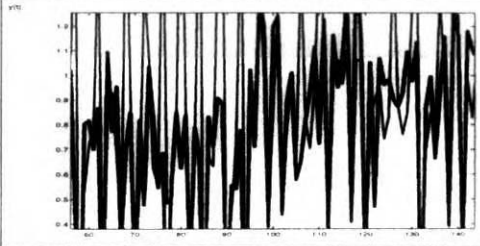
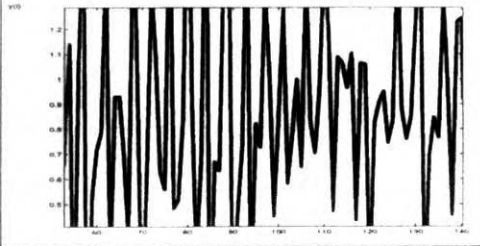
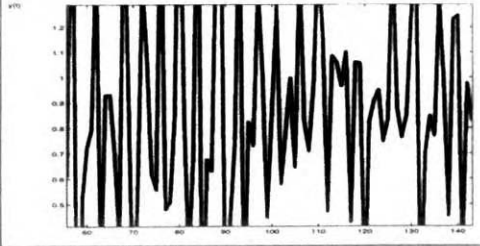
Replacement	MSE	Figure
Series N=16.	0.0415273	
Series N=20.	$9.43e-7$	
Simplified	$9.96849e-007$	

Table 6: Neural network MLP architecture 15,1. Comparison of networks with original activation function and replacements derived for the *exp()* function.(Output time responses)

Network architecture	error	Notice	Eval. time[%]
MLP 15,1 original	$4.28 \cdot 10^{-7}$	Goal met	100
MLP 15,1, simpl. approx.	-	-	20
MLP 10,1 simpl. approx.	0.069	MU reached, Goal does not	13
MLP 5,1 simpl. approx.	0.0689	MU reached, Goal does not	6.6
MLP 1,1 simpl. approx.	$9.9 \cdot 10^{-7}$	Goal met	1.5

Table 7: Computation complexity of the MLP network optimized by the structure optimization algorithm for problem (1)

Optimization level	optimization	optimization	approx. speed-up
Data preprocessing	strong	-	-
Local observer	strong	deterministic	-
Structure optimization	weak	nondeterministic	0-100%
Data overhead reduction	strong	deterministic	0-10%
Function call overhead reduction	strong	deterministic	0-10%
Elementary function overhead	strong,weak	deterministic	0-300%
Linear code dependence reduction	strong	deterministic	0-10%

Table 8: Evaluation of results obtained at different optimization levels and their evaluation.

3 Case study.

We consider the following non-linear system described by a difference equation:

$$y(k) = \frac{y(k-1)y(k-2)y(k-2)y(k-3)u(k-3) - y(k-1)y(k-2)u(k-3) + u(k-4)}{1 + y^2(k-2) + y^2(k-3)} \quad (1)$$

The system has behavior of unstable system in neighborhood point (2) as was shown in [1]. For the sake of simplicity the main system features are summed-up here:

- Non linear system.
- Unstable on response greater than 2.

The system was modelled by standard MLP network with the following architecture:

- 10 inputs, with order of input = 5 and order of output = 5.
- 1st hidden layer consisting of 15 neurons with *tansig* function used.
- 2nd layer with linear identity function.

The error obtained by *LM* training algorithm was $4.28 \cdot 10^{-7}$ on training set after 300 cycles. The obtained results are stored in the table (3). Next, the results obtained in each optimization levels are shown in the following tables: Table Tab. 1 and Tab. 2 compares the computational complexity of original *exp* function and replacement. The possible optimization results for data access and function call access reduction are shown in table Tab. 4 and Tab. 5. Final possible obtained optimization by each level is shown in table Tab. 8. On the following figures Tab. 7 - Tab. 9 the detailed view on the simulation results compared with the original network MLP 15,1 (15,1 means 15 neurons in the first hidden layer and 1 in the second layer) is shown. The simulations were performed with different activation function replacement (finite reduction of series for $N=2,4,8,16,20$, simple polynomial approximation and simplified approximation. The simulation results for MLP optimized by algorithm of structure optimization are stored in the Tab. 10 - Tab. 12. The simplified approximation was used as approximation technique with the best results. As it can be viewed, only the MLP1,1 has bad results (we notice, that simulation was done without affecting network learning algorithm, such as its parameters, thus some learning processes were not finished).

Acknowledgements

This paper was partially supported by the Slovak Scientific Grant Agency, Grant no. 1/7630/20.

References

- [1] Čapkovič Miroslav “Neural Networks For Real Control Time Systems, Large And Large Scaled Problems”, *Thesis*, Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, 2002
- [2] Shmuel Gal and Boris Bachelis “Accurate Elementary Mathematical Library for the IEEE Floating Point Standard”, “ACM Transactions on Mathematical Software, vol.17, no.1, p.26–45, mar. 1991”

Composite Fuzzy Measure and its Application to Investment Decision Making

Toshihiro KAINO^{*}, Kaoru HIROTA^{**}

^{*}*School of Business Administration, Aoyama Gakuin University,*

4-4-25, Shibuya, Shibuya-ku, Tokyo, Japan 150-8366. kain@mue.biglobe.ne.jp

^{**}*Department of Computational Intelligence and Systems Science, Interdisciplinary
Graduate School of Science and Engineering, Tokyo Institute of Technology, Midori-ku,
Yokohama, Japan 226-8502.*

hirota@hrt.dis.titech.ac.jp

Abstract. In the application using fuzzy measure (on real numbers), it is a problem how to evaluate the inbetween intervals each of which is characterized by a fuzzy measure. Especially, when the differentiation of the Choquet integral is applied to the real world problem, this is essentially important. So, a composite fuzzy measure built up from fuzzy measures defined on fuzzy measurable spaces has been proposed using composite fuzzy weights by Kaino and Hirota, where the measurable space of this composite fuzzy measure is the direct sum of measurable spaces. Here, an associative, composite fuzzy measure built up from three fuzzy measures is proposed in a constructive way. As an application, it is applied to the automobile factory capital investment decision making problem. It is assumed that an automobile company has a sales plan of a new car. The current factory line has a capacity to manufacture 3,200 new cars, additional to the current car lines. Then, by the use of this composite fuzzy measure, the differentiation of the Choquet integral becomes a quantitative index for decision making, which is confirmed by this decision making experiment.

Keywords: Composite Fuzzy Measure, Differentiation of Choquet Integral, Decision Making

1. Introduction

The concept of fuzzy measure and fuzzy integral has been introduced by Sugeno[1] and the functional (the Choquet integral) defined by Choquet[2] is revalued in terms of fuzzy measure. These fuzzy integrals have been applied to multi-criteria evaluation and prediction fields [3], [4], [5]. But, differentiation as contrast concept of integral has not been studied enough except differentiation of fuzzy functions[6] and fuzzy relations[7]. Kaino and Hirota[8],[10] proposed differentiation of the Choquet integral of a measurable function over a real fuzzy measure space. Moreover, differentiation of the X axis real interval limited Choquet integral over a real fuzzy measure space is proposed by the limitation process of a fuzzy measure shift[9][11]. By the way, in the application using fuzzy measure on a real number, it is a problem how to evaluate the inside and inbetween intervals. So, a composite fuzzy measure built up from two fuzzy measures using composite fuzzy weights is proposed by Kaino and Hirota[12]. This is proved on the proposition that a composite

measure built up from two fuzzy measures will be a fuzzy measure, and it is extended to a composite measure built up from three fuzzy measures in a theorem. But, how to construct it has not been indicated? Here, a direct construction method is proposed, where weight like three $[0,1]$ s are introduced and they will be assigned by the user in each application. As an application, the automobile factory capital investment decision making problem is mentioned to show how to apply the proposed construction method.

2. Composite Fuzzy Measure Built up from Fuzzy Measures

In the application using fuzzy measure on a real number, it is a problem how to evaluate the inside and inbetween intervals which were characterized by fuzzy measures. Then, a composite fuzzy measure built up from two fuzzy measures using composite fuzzy weights is proposed by Kaino and Hirota[12]. Firstly, a composite measure is shown, where each interval is evaluated by a fuzzy measure.

Before formally proposing a composite fuzzy measure, it is shown that a composite measurable space using direct sum of two measurable spaces is a measurable space as *Proposition I*.

Proposition I. $(X_1 \oplus X_2, F_1 \oplus F_2)$ is a measurable space, where $F_1 \oplus F_2 = \{E_1 \oplus E_2 \mid E_i \in F_i (i=1,2)\}$, and \oplus indicates a direct sum. (1)

In the same way, it is proved that a composite measurable space using direct sum of some measurable spaces is a measurable space by Kaino and Hirota[12].

Proposition II. $(\bigoplus_{i=1}^n X_i, \bigoplus_{i=1}^n F_i)$ is a measurable space,

where $\bigoplus_{i=1}^n F_i = \{ \bigoplus_{i=1}^n E_i \mid E_i \in F_i (i=1,2,\dots,n) \}$. (Let \oplus be a direct sum) (2)

Now, a composite fuzzy measure built up from two fuzzy measures on the direct sum of two fuzzy measure spaces is given as *Theorem I*.

Theorem I. Let (X_i, F_i, μ_i) ($i=1,2$) be fuzzy measure spaces. Then $(X_1 \oplus X_2, F_1 \oplus F_2, \mu_{1 \oplus 2})$ is a fuzzy measure space, where $\forall E = E_1 \oplus E_2 \in F_1 \oplus F_2$, $\lambda_1, \lambda_2 \in [0,1]$, $\mu_{1 \oplus 2}(E_1 \oplus E_2) = \lambda_1 \mu_1(E_1) + \lambda_2 \mu_2(E_2) + (1 - \lambda_1 - \lambda_2) \mu_1(E_1) \mu_2(E_2)$. (3)

Moreover, a composite measure built up from two fuzzy measures is recursively extended to a composite measure built up from a finite number of fuzzy measures by Kaino and Hirota[12]

Theorem II. Let (X_i, F_i, μ_i) ($i=1,2,\dots,n$) be fuzzy measure spaces. Then $((\bigoplus_{i=1}^n (X_i \oplus X_2 \oplus X_3) \oplus X_n) \oplus ((F_1 \oplus F_2) \oplus F_3) \oplus F_n, \mu_{((\bigoplus_{i=1}^n (X_i \oplus X_2 \oplus X_3) \oplus X_n) \oplus ((F_1 \oplus F_2) \oplus F_3) \oplus F_n)})$ is a fuzzy measure space, where $\forall ((\bigoplus_{i=1}^n (X_i \oplus X_2 \oplus X_3) \oplus X_n) \oplus ((F_1 \oplus F_2) \oplus F_3) \oplus F_n) \in ((F_1 \oplus F_2) \oplus F_3) \oplus F_n$, $\lambda_1, \lambda_2, \lambda_3, \dots, \lambda_n \in [0,1]$, $\mu_{((\bigoplus_{i=1}^n (X_i \oplus X_2 \oplus X_3) \oplus X_n) \oplus ((F_1 \oplus F_2) \oplus F_3) \oplus F_n)} = \lambda_1 \mu_1(E_1) + \lambda_2 \mu_2(E_2) + \lambda_3 \mu_3(E_3) + \dots + \lambda_n \mu_n(E_n) + (1 - \lambda_1 - \lambda_2 - \lambda_3 - \dots - \lambda_n) \mu_{((\bigoplus_{i=1}^n (X_i \oplus X_2 \oplus X_3) \oplus X_n) \oplus ((F_1 \oplus F_2) \oplus F_3) \oplus F_n)}$. (4)

And, the associative, composite fuzzy measure built up from three fuzzy measures is introduced by Kaino and Hirota[12].

Theorem III. Let (X_i, F_i, μ_i) ($i=1,2,3$) be fuzzy measure spaces. Let $(X_1 \oplus X_2, F_1 \oplus F_2, \mu_{1 \oplus 2})$ be a fuzzy measure space as shown in *Theorem I*. And, let $((X_1 \oplus X_2) \oplus X_3, (F_1 \oplus F_2) \oplus F_3, \mu_{(1 \oplus 2) \oplus 3})$ be a fuzzy measure space, where $\forall E = (E_1 \oplus E_2) \oplus E_3 \in (F_1 \oplus F_2) \oplus F_3$,

$$\begin{aligned} \forall \lambda_{1 \oplus 2}, \lambda_1, \lambda_2, \lambda_3 \in (0,1], \lambda_1' = \frac{\lambda_1}{\lambda_{1 \oplus 2}}, \lambda_2' = \frac{\lambda_2}{\lambda_{1 \oplus 2}}, \lambda_3' = \lambda_3, A = \frac{1 - \lambda_1' - \lambda_2'}{\lambda_1' \lambda_2' \lambda_{1 \oplus 2}}, \\ \mu_{(1 \oplus 2) \oplus 3}((E_1 \oplus E_2) \oplus E_3) = \lambda_1' \mu_1(E_1) + \lambda_2' \mu_2(E_2) + \lambda_3' \mu_3(E_3) \\ + A(\lambda_1' \lambda_2' \mu_1(E_1) \mu_2(E_2) + \lambda_2' \lambda_3' \mu_2(E_2) \mu_3(E_3) + \lambda_1' \lambda_3' \mu_1(E_1) \mu_3(E_3)) \\ + A^2 \lambda_1' \lambda_2' \lambda_3' \mu_1(E_1) \mu_2(E_2) \mu_3(E_3). \end{aligned} \quad (5)$$

Then this construction process does not depend on the order, i.e.,

$$\mu_{(1 \oplus 2) \oplus 3}((E_1 \oplus E_2) \oplus E_3) = \mu_{1 \oplus 2 \oplus 3}(E_1 \oplus E_2 \oplus E_3). \quad (6)$$

Here, Theorem III will be arranged to Theorem IV so that it will be easy to use.

Theorem IV. Let (X_i, F_i, μ_i) ($i=1,2,3$) be fuzzy measure spaces.

Then $(X_1 \oplus X_2 \oplus X_3, F_1 \oplus F_2 \oplus F_3, \mu_{1 \oplus 2 \oplus 3})$ is a fuzzy measure space, where

$$\begin{aligned} \mu_{1 \oplus 2 \oplus 3}(E_1 \oplus E_2 \oplus E_3) &= \lambda_1 \mu_1(E_1) + \lambda_2 \mu_2(E_2) + \lambda_3 \mu_3(E_3) \\ &+ A(\lambda_1 \lambda_2 \mu_1(E_1) \mu_2(E_2) + \lambda_2 \lambda_3 \mu_2(E_2) \mu_3(E_3) + \lambda_3 \lambda_1 \mu_3(E_3) \mu_1(E_1)) \\ &+ A^2 \lambda_1 \lambda_2 \lambda_3 \mu_1(E_1) \mu_2(E_2) \mu_3(E_3), \end{aligned} \quad (7)$$

for $\forall E_1 \oplus E_2 \oplus E_3 \in F_1 \oplus F_2 \oplus F_3$, and $\forall \lambda_1, \lambda_2, \lambda_3 \in [0, 1]$, such that

$$\lambda_1 \lambda_2 \lambda_3 A^2 + (\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1)A + \lambda_1 + \lambda_2 + \lambda_3 = 1, \quad (8)$$

holds true and A is the greater solution of (8).

(proof)

(i) The discriminant of equation (8),

$$\begin{aligned} D &= (\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1)^2 - 4 \lambda_1 \lambda_2 \lambda_3 (\lambda_1 + \lambda_2 + \lambda_3 - 1) = \lambda_1^2 \lambda_2^2 + \lambda_2^2 \lambda_3^2 \\ &+ \lambda_3^2 \lambda_1^2 + 2 \lambda_1 \lambda_2 \lambda_3 (2 \square \lambda_1 + \lambda_2 + \lambda_3) \square f(\lambda_1, \lambda_2, \lambda_3). \end{aligned} \quad (9)$$

$$\begin{aligned} \frac{\partial f}{\partial \lambda_1} &= 2 \lambda_1 (\lambda_2^2 + \lambda_3^2) + 2 \lambda_2 \lambda_3 (2 \square \lambda_1 \square \lambda_2 \square \lambda_3) \square 2 \lambda_1 \lambda_2 \lambda_3 \\ &= 2 \lambda_2^2 (\lambda_1 - \lambda_3) + 2 \lambda_3^2 (\lambda_1 - \lambda_2) + 4 \lambda_2 \lambda_3 (1 - \lambda_1). \end{aligned} \quad (10)$$

$$\text{If } \lambda_1 \geq \lambda_3, \lambda_1 \geq \lambda_2, \text{ then } \frac{\partial f}{\partial \lambda_1} \geq 0. \therefore f \geq 0. \quad (11)$$

$$\text{Similarly, if } \lambda_2 \geq \lambda_3, \lambda_2 \geq \lambda_1, \text{ then } \frac{\partial f}{\partial \lambda_2} \geq 0. \therefore f \geq 0. \quad (12)$$

$$\text{If } \lambda_3 \geq \lambda_2, \lambda_3 \geq \lambda_1, \text{ then } \frac{\partial f}{\partial \lambda_3} \geq 0. \therefore f \geq 0. \quad (13)$$

Therefore, $D \geq 0$. (14)

And, it should be noticed that, in the case of $\lambda_1 \lambda_2 \lambda_3 \neq 0$, the greater solution A_+ of

$$\text{equation (8), } A_+ = \frac{-(\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1) + \sqrt{D}}{2 \lambda_1 \lambda_2 \lambda_3} \geq -1, \quad (15)$$

because $(1 - \lambda_1)(1 - \lambda_2)(1 - \lambda_3) \geq 0$, $1 + \lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1 - \lambda_1 - \lambda_2 - \lambda_3 - \lambda_1 \lambda_2 \lambda_3 \geq 0$,

$$4 \lambda_1 \lambda_2 \lambda_3 (1 + \lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1 - \lambda_1 - \lambda_2 - \lambda_3 - \lambda_1 \lambda_2 \lambda_3) \geq 0,$$

$$-4 \lambda_1 \lambda_2 \lambda_3 (\lambda_1 + \lambda_2 + \lambda_3 - 1) + 4 \lambda_1 \lambda_2 \lambda_3 (\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1) - 4 \lambda_1^2 \lambda_2^2 \lambda_3^2 \geq 0,$$

$$(\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1)^2 - 4 \lambda_1 \lambda_2 \lambda_3 (\lambda_1 + \lambda_2 + \lambda_3 - 1) \geq$$

$$(\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1)^2 - 4 \lambda_1 \lambda_2 \lambda_3 (\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1) + 4 \lambda_1^2 \lambda_2^2 \lambda_3^2,$$

$$\sqrt{(\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1)^2 - 4 \lambda_1 \lambda_2 \lambda_3 (\lambda_1 + \lambda_2 + \lambda_3 - 1)} \geq (\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1) - 2 \lambda_1 \lambda_2 \lambda_3,$$

$$\frac{-(\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1) + \sqrt{D}}{2 \lambda_1 \lambda_2 \lambda_3} \geq -1. \quad (16)$$

$$\begin{aligned} (\square) \mu_{1 \oplus 2 \oplus 3}(X_1 \oplus X_2 \oplus X_3) &= \lambda_1 \mu_1(X_1) + \lambda_2 \mu_2(X_2) + \lambda_3 \mu_3(X_3) \\ &+ A(\lambda_1 \lambda_2 \mu_1(X_1) \mu_2(X_2) + \lambda_2 \lambda_3 \mu_2(X_2) \mu_3(X_3) + \lambda_3 \lambda_1 \mu_3(X_3) \mu_1(X_1)) \\ &+ A^2 \lambda_1 \lambda_2 \lambda_3 \mu_1(X_1) \mu_2(X_2) \mu_3(X_3) \\ &= \lambda_1 + \lambda_2 + \lambda_3 + A(\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_3 \lambda_1) + A^2 \lambda_1 \lambda_2 \lambda_3 = 1. \end{aligned} \quad (17)$$

(iii) monotonicity. $\forall E, \forall F \in F_1 \oplus F_2 \oplus F_3, E \subset F, \exists E_i, F_i \in F_i$ ($i=1,2,3$).

Here, put $\mu_i(F_i) \Rightarrow f_i, \mu_i(E_i) \Rightarrow e_i, (i=1,2,3)$.

$$\begin{aligned} \mu_{1 \oplus 2 \oplus 3}(F_1 \oplus F_2 \oplus F_3) - \mu_{1 \oplus 2 \oplus 3}(E_1 \oplus E_2 \oplus E_3) &= \lambda_1(f_1 - e_1) + \lambda_2(f_2 - e_2) + \lambda_3(f_3 - e_3) + A\{\lambda_1 \lambda_2(f_1 f_2 - e_1 e_2) + \lambda_2 \lambda_3(f_2 f_3 - e_2 e_3) \\ &+ \lambda_3 \lambda_1(f_3 f_1 - e_3 e_1)\} + A^2 \lambda_1 \lambda_2 \lambda_3 (f_1 f_2 f_3 - e_1 e_2 e_3) \\ &= \lambda_1(f_1 - e_1) + \lambda_2(f_2 - e_2) + \lambda_3(f_3 - e_3) + A\{\lambda_1 \lambda_2(f_1 f_2 - e_1 f_2 + e_1 f_2 - e_1 e_2) \\ &+ \lambda_2 \lambda_3(f_2 f_3 - f_2 e_3 + f_2 e_3 - e_2 e_3) + \lambda_3 \lambda_1(f_3 f_1 - f_3 e_1 + f_3 e_1 - e_3 e_1)\} \end{aligned}$$

$$\begin{aligned}
& + A^2 \lambda_1 \lambda_2 \lambda_3 (f_1 f_2 f_3 - f_1 f_2 e_3 + f_1 f_2 e_3 - e_1 e_2 e_3) \\
& = \lambda_1 (f_1 - e_1) (1 + A \lambda_2 f_2) (1 + A \lambda_3 f_3) + \lambda_2 (f_2 - e_2) (1 + A \lambda_1 e_1) (1 + A \lambda_3 e_3) \\
& + \lambda_3 (f_3 - e_3) (1 + A \lambda_1 e_1) (1 + A \lambda_2 f_2) \geq 0. \quad (\because A \geq -1)
\end{aligned}
\tag{18}$$

(q.e.d)

Remark I. In the case of $\lambda_3=0$, $\lambda_1 \lambda_2 \neq 0$, (7) becomes

$$\mu_{1 \oplus 2}(E_1 \oplus E_2) = \lambda_1 \mu_1(E_1) + \lambda_2 \mu_2(E_2) + (1 \square \lambda_1 \square \lambda_2) \mu_1(E_1) \mu_2(E_2). \tag{19}$$

Theorem IV is reduced to Theorem I.

Remark II. In the case of $\lambda_2 = \lambda_3 = 0$, then (7) becomes $\lambda_1 = 1$, essentially, single fuzzy measure space.

Remark III. In the case of $\lambda_1 = \lambda_2 = \lambda_3 = \lambda$, then (7) becomes

$$\begin{aligned}
& \mu_{1 \oplus 2 \oplus 3}(E_1 \oplus E_2 \oplus E_3) = \lambda (\mu_1(E_1) + \mu_2(E_2) + \mu_3(E_3)) \\
& + \frac{\sqrt{4\lambda - 3\lambda^2} - 3\lambda}{2} (\mu_1(E_1) \mu_2(E_2) + \mu_2(E_2) \mu_3(E_3) + \mu_3(E_3) \mu_1(E_1)) + \frac{2 + 3\lambda - 3\sqrt{4\lambda - 3\lambda^2}}{2} \mu_1(E_1) \mu_2(E_2) \mu_3(E_3)
\end{aligned}
\tag{20}$$

for $\forall E_1 \oplus E_2 \oplus E_3 \in \mathbf{F}_1 \oplus \mathbf{F}_2 \oplus \mathbf{F}_3$, and $\forall \lambda \in [0, 1]$. Especially if $\lambda = 0, \frac{1}{3}, 1$, then

$$\mu_{1 \oplus 2 \oplus 3}(E_1 \oplus E_2 \oplus E_3) = \begin{cases} \mu_1(E_1) \mu_2(E_2) \mu_3(E_3) & (\lambda = 0) \\ \frac{\mu_1(E_1) + \mu_2(E_2) + \mu_3(E_3)}{3} & (\lambda = \frac{1}{3}) \\ \mu_1(E_1) + \mu_2(E_2) + \mu_3(E_3) - (\mu_1(E_1) \mu_2(E_2) + \mu_2(E_2) \mu_3(E_3) + \mu_3(E_3) \mu_1(E_1)) + \mu_1(E_1) \mu_2(E_2) \mu_3(E_3) & (\lambda = 1) \end{cases} \tag{21}$$

3. Application to Capital Investment Decision Making Problem

The composite fuzzy measure built up from three fuzzy measures is applied to the automobile factory capital investment decision making problem. An automobile company has a sales plan of a new car. The current factory line has a capacity to manufacture 3,200 new cars, additional to the current car lines. So, if that company has a sales plan to sell over 3,200 new cars annually, they must invest into the new factory. Some marketing staffs evaluate the possibility of the annual sales amount of the new car as the following table from the results.

Amount	[1000,200 0)	[2000,300 0)	[3000,4000)
Evaluation n	0.3	0.6	0.4

Table 1. Annual sales estimate of a new car

Now, the evaluation of the inside of the each interval is given by an additive measure, i.e., a probability measure, and the evaluation of the inbetween intervals is given by a fuzzy measure. Let $X_1=[1000, 2000)$, $X_2=[2000, 3000)$, and $X_3=[3000, 4000)$. Let (X_i, \mathbf{F}_i, P_i) ($i=1,2,3$) be probability spaces (which are a kind of fuzzy measure spaces). Then, $(\bigoplus_{i=1}^3 X_i, \sigma(\{F_i\}_{i=1}^3), \mu_{1 \oplus 2 \oplus 3})$ will be a composite fuzzy measure of probabilities P_i ($i=1,2,3$) (cf.

Theorem IV). Then, suppose that λ_i ($i=1,2,3$) are the values from Table 1, i.e., $\lambda_1=0.3$, $\lambda_2=0.6$, $\lambda_3=0.4$. (22)

Here, $\mu_{1 \oplus 2 \oplus 3}([1000, x])$ ($\forall x \in \bigoplus_{i=1}^3 X_i$) is calculated as :

$$i) \quad 1000 \leq x < 2000: \mu_{1 \oplus 2 \oplus 3}([1000, x]) = \lambda_1 P_1([1000, x]) + \lambda_2 P_2(\phi_2) + \lambda_3 P_3(\phi_3) \\ + A(\lambda_1 \lambda_2 P_1([1000, x]) P_2(\phi_2) + \lambda_2 \lambda_3 P_2(\phi_2) P_3(\phi_3) + \lambda_1 \lambda_3 P_1([1000, x]) P_3(E_3)) \\ + A^2 P_1([1000, x]) P_2(E_2) P_3(E_3) = \lambda_1 P_1([1000, x]) = \lambda_1 \frac{x-1000}{1000}. \quad (23)$$

$$ii) \quad 2000 \leq x < 3000: \mu_{1 \oplus 2 \oplus 3}([1000, x]) = \lambda_1 + (\lambda_2 + A \lambda_1 \lambda_2) \frac{x-2000}{1000}. \quad (24)$$

$$iii) \quad 3000 \leq x < 4000: \mu_{1 \oplus 2 \oplus 3}([1000, x]) \\ = \lambda_1 + \lambda_2 + A \lambda_1 \lambda_2 + (1 + A \lambda_2 \lambda_3 + A \lambda_1 \lambda_3 + A^2 \lambda_1 \lambda_2 \lambda_3) \frac{x-3000}{1000}. \quad (25)$$

Using the equations (15),

$$A_+ = \frac{-(\lambda_1 \lambda_2 + \lambda_2 \lambda_3 + \lambda_1 \lambda_3) + \sqrt{D}}{2 \lambda_1 \lambda_2 \lambda_3} = -0.604235652. \quad (26)$$

Here, it is assumed that the sales amount (million ¥) of a new car will be given as the following equation (27) after consideration of discount,

$$f(x) = \begin{cases} x & (1000 \leq x < 2000) \\ 0.9(x-2000)+2000 & (2000 \leq x < 3000) \\ 0.8(x-3000)+2900 & (3000 \leq x < 4000) \end{cases} \quad (27)$$

The expected sales revenue of this new car will be derived from the X axis $[1000, x]$ limited Choquet integral[9],[11] $F_X(x) \cdot F_X([1000, x])$ as follows:

$$i) \quad 1000 \leq x < 2000$$

$$F_X(x) = \int_0^x \mu_{1 \oplus 2 \oplus 3}(\{t \mid f(t) \geq t\} \cap [1000, x] \cap X) dt \\ = \int_0^{1000} \mu_{1 \oplus 2 \oplus 3}([1000, x]) dt + \int_{1000}^x \mu_{1 \oplus 2 \oplus 3}([t, x]) dt = 0.00015x^2 - 150. \quad (28)$$

$$ii) \quad 2000 \leq x < 3000$$

$$F_X(x) = \int_0^{1000} \mu_{1 \oplus 2 \oplus 3}([1000, x]) dt + \int_{1000}^{2000} \mu_{1 \oplus 2 \oplus 3}([t, x]) dt + \int_{2000}^x \mu_{1 \oplus 2 \oplus 3} \left(\left[\frac{10t-2000}{9}, x \right] \right) dt \\ = 0.00027x^2 - 0.043143626x - 543.7127479. \quad (29)$$

$$iii) \quad 3000 \leq x < 4000$$

$$F_X(x) = \int_0^{1000} \mu_{1 \oplus 2 \oplus 3}([1000, x]) dt + \int_{1000}^{2000} \mu_{1 \oplus 2 \oplus 3}([t, x]) dt + \int_{2000}^{2900} \mu_{1 \oplus 2 \oplus 3} \left(\left[\frac{10t-2000}{9}, x \right] \right) dt \\ + \int_{2900}^{0.8x+500} \mu_{1 \oplus 2 \oplus 3} \left(\left[\frac{10t-5000}{8}, x \right] \right) dt = 0.00016x^2 - 0.224622103x + 990.7226818. \quad (30)$$

The differentiation of the X axis $[1000, x]$ limited Choquet integral[9],[11] $F_X'(x)$ will be given as

$$F_X'(x) = \begin{cases} 0.0003x & (1000 \leq x < 2000) \\ 0.00054x - 0.043143626 & (2000 \leq x < 3000) \\ 0.00032x - 0.224622103 & (3000 \leq x < 4000) \end{cases} \quad (31)$$

The differentiation of the X axis $[1000, x]$ limited Choquet integral using a composite fuzzy measure can be considered to be the increase or decrease rate of the expected sales amount per a car. Now, it is assumed that the cost of each new car is given as shown in the fig.1 because of the additional automobile factory capital investment for increasing the products excess of 3,200 cars annually. Then, it is easy to make a decision to invest or not using the differentiation of the X axis real interval limited Choquet integral using composite fuzzy measure over X , which gives us the increase or decrease of the expected sales revenue (also profit) per a new car. Recently, ROI (Return On Investment) becomes the important index for a capital investment decision making. This differentiation using composite fuzzy measure is confirmed to be a useful tool for this decision making.

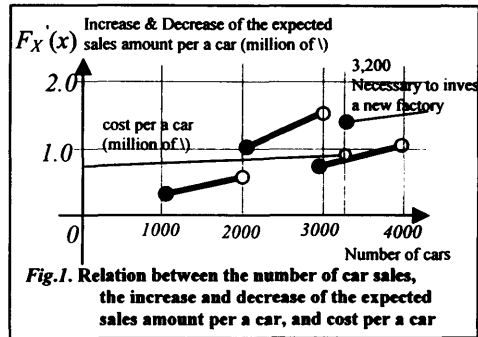


Fig.1. Relation between the number of car sales, the increase and decrease of the expected sales amount per a car, and cost per a car

4. Conclusions

An associative, composite fuzzy measure built up from three fuzzy measures is proposed using composite fuzzy weights, and it is applied to the automobile factory capital investment decision making problem. It is proved that a composite measure built up from two fuzzy measures using composite fuzzy weights will be a fuzzy measure, and recursively extended to a composite fuzzy measure built up from plural measurable spaces. And the associative, composite fuzzy measure built up from three fuzzy measures is introduced. Here, it is arranged to a constructive theorem and this composite fuzzy measure is applied to the automobile factory capital investment decision making problem. It is assumed that an automobile company has a sales plan of a new car. The current factory line has a capacity to manufacture 3,200 new cars, additional to the current car lines. It is also assumed that the cost of each new car goes up, because of the additional automobile factory capital investment for increasing the products excess of 3,200 cars annually. Then, it is easy to make a decision to invest or not using the differentiation of the X axis real interval limited Choquet integral using this composite fuzzy measure, which gives us the increase or decrease of the expected sales revenue (also profit) per a new car. By the use of this composite fuzzy measure, the differentiation of the Choquet integral becomes a quantitative index for decision making, and it is confirmed to be a tool for this decision making. After this, to study about composite weights, the associative, composite fuzzy measure built up from a finite number of fuzzy measures, and its application to a real world are left for the future studies.

References

- [1] M. Sugeno, "Fuzzy Measure and Fuzzy Integral", *Journal of the Society of Instrument and Control Engineers*, Vol. 8, No. 2, 1972, pp. 218-226. (in Japanese language)
- [2] G. Choquet, "Theory of capacities", *Ann. Inst. Fourier* 5, 1953, pp. 131-295.
- [3] M. Higuchi, "Fuzzy Integral application to estimate of grindstone's quality-Study of grinding sound (Report 4.)", *Journal of the Japan Society for Precision Engineering*, Vol. 54, No. 4, 1988, pp. 761-766.
- [4] K. Gotoh, Y. Toriyama, and O. Itoh, "Fuzzy Memory Based Reasoning and Total Evaluation of Plant Status by Using Fuzzy Measure", *Journal of Japan Society for Fuzzy Theory and Systems*, 7, 1995, pp. 390-401.
- [5] K. Gotoh, Y. Toriyama, and O. Itoh : "Intelligent Alarm Method by Fuzzy Measure and Its Application to Plant Abnormality Prediction", *Proc. of FUZZ-IEEE/IFES '95, Yokohama, 1995*, pp.395-400.

- [6] M. Mizumoto, *Fuzzy set theory and its application*, SAIENSU-SHA Co., Ltd., 1987, pp. 149-152. (in Japanese language)
- [7] T. Kaino, K. Hirota: "Derivative of Fuzzy Relations and Its Application to Capital Investment Decision Making Problem", *Proc. of IFSA'99, Taiwan, 1999*, pp. 995-998, 1999.
- [8] T. Kaino, K. Hirota: "Differentiation of the Choquet Integral of a Nonnegative Measurable Function", *Proc. of FUZZ-IEEE'99, Seoul, 1999*, Vol. III, pp. 1322-1327.
- [9] T. Kaino, K. Hirota, "Differentiation of Nonnegative Measurable Function Choquet Integral over Real Fuzzy Measure Space and Its Application to Financial Option Trading Model", *Proc. of IEEE SMC'99, Tokyo, 1999*, Vol. III, pp. 73-78.
- [10] T. Kaino, K. Hirota, "Differentiation of Choquet Integral for Measurable Function and its Application to Long Term Credit Rating", *Journal of Advanced Computational Intelligence*, Vol. 4, No. 1, pp. 66-75, 2000.
- [11] T. Kaino, K. Hirota, "Differentiation of Choquet Integral for Nonnegative Measurable Function and its Application to Capital Investment Decision Making", *Proc. of FUZZ-IEEE'99, Texas, 2000*, pp. 89-93.
- [12] T. Kaino, K. Hirota, "Composite Fuzzy Measure and its Application to Decision Making", *Proc. of IFSA2001, Vancouver, 2001*, pp. 2551-2555.

Prediction of the Index Fund on the Basis of Fuzzy Inference Systems

Vladimír OLEJ, Marián FEDURCO

*Matej Bel University, School of Finance, Department of Information Systems, Tajovského
10, 974 01 Banská Bystrica, Slovak Republic
olej@financ.umb.sk, fedurco@financ.umb.sk*

Abstract. This paper presents (on the basis of analysis of passive investment strategies) the design of the fuzzy inference system of the Sugeno type. By means of this fuzzy inference system the investor is able to predict the closing price of the index fund.

Keywords: *Fuzzy inference system of the Sugeno type, prediction of the index fund, the indicators of technical analysis.*

1. Introduction

Decision-making processes and control systems on the basis of fuzzy inference systems (FIS) [1 to 5] are moving from technical scientific disciplines to non-technical spheres [3,4,5]. An example of this is the prediction of price development of passively constructed portfolios in the economic area. The basic types of the investment strategies used in the capital markets can be defined on the basis of methodological analysis and approach analysis of the investment strategies. As an example we can mention the construction of the index fund, so that fund returns copy the development of market portfolio returns. The index fund can be modelled by means of FIS, which enables the prediction of price development of the index fund in time $t + \Delta t$. The indicators of technical analysis of securities are used as the inputs in the trading system designed on the basis of FIS. Two basic types of FISs can be designed on the basis of the general structure of FIS [1], i.e. the Mamdani and Sugeno types. Both types of FISs differ in determination of the outputs. Different formulation of the outputs results in different construction of the inference rules (IR). The user can construct IRs (on the basis of his own experience) or IRs can be obtained by extraction from the historical data. Fuzzification of the input variables and application of operators in IRs is the same in both types of FISs. The usage of FIS of the Sugeno type increases the efficiency of the defuzzification process, because the computational process is considerably reduced in comparison with more general FIS of the Mamdani type.

2. Fuzzy Inference Systems

The general structure of FIS [1] contains the fuzzification process by means of input membership functions, construction of base of inference rules (BIR) or automatic extraction of IRs from the input data, application of operators (AND, OR, NOT) in IRs, implication and aggregation within IRs and the defuzzification process of obtained outputs to the crisp values. Normalisation of the inputs and their transformation to the range of values of the input membership functions is realised during the fuzzification process. The inference mechanism is based on the operations of fuzzy logic and implication within IRs [1 to 5].

Transformation of the outputs of individual IRs to the output fuzzy set is realised on the basis of the aggregation process. Conversion of fuzzy values to expected crisp values is realised during the defuzzification process. The most commonly-used defuzzification method is the Centre of Gravity method, one of the simplest defuzzification methods is the Max Criterion Method, eventually the Mean of Maxima Method. There exists no universal method for designing shape, the number and parameters of the input and output membership functions. Triangular, trapezoidal and other membership functions are used for the design of FIS. The input to the fuzzification process is the crisp value given by universe of discourse (reference set). The output of the fuzzification process is the membership function value.

The construction of BIRs can be realised by extraction of IRs from historical data, if they are at one's disposal. Various optimisation methods for number of IRs in BIRs are presented in [1 to 5]. One of the possibilities for optimising BIRs from historical data is the so-called Adaptive Neuro - Fuzzy Inference System method (ANFIS method). The essence of this method is the neuro - adaptive learning process, on the basis of which it is possible to derive the parameters of membership functions and to extract BIRs. The input data are mapped to the output data by means of this method, whereas parameters of individual membership functions are gradually changed during the learning process, so that relations between the space of the input variables and the output variables are described in the best possible way. The back-propagation method [1], a combination of the back-propagation method with the least squares method and finally evolution stochastic optimisation algorithms [6] can be used in the learning process. The BIRs consists of IF – THEN IRs [1 to 5]. Inference rules are used for the construction of conditional terms, which create the basis of FIS.

Let $x_1, x_2, \dots, x_i, \dots, x_n$ be the input variables defined in the reference sets $X_1, X_2, \dots, X_i, \dots, X_n$ and let y be the output variable defined in the reference set Y . Then FIS has n input variables and one output variable. Every set $X_i, i = 1, \dots, n$, can be divided into $p_j, j = 1, \dots, m$, the fuzzy sets $\mu_1^{(i)}(x), \mu_2^{(i)}(x), \dots, \mu_{pj}^{(i)}(x), \dots, \mu_m^{(i)}(x)$. The individual fuzzy sets $\mu_1^{(i)}(x), \mu_2^{(i)}(x), \dots, \mu_{pj}^{(i)}(x), \dots, \mu_m^{(i)}(x), i = 1, \dots, n; j = 1, \dots, m$ represent assignment of linguistic variables relating to sets X_i . The set Y is also divided into $p_k, k = 1, \dots, o$, the fuzzy sets $\mu_1(y), \mu_2(y), \dots, \mu_{pk}(y), \dots, \mu_o(y)$. The fuzzy sets $\mu_1(y), \mu_2(y), \dots, \mu_{pk}(y), \dots, \mu_o(y)$ represent assignment of linguistic variables for the set Y . Then IF – THEN IR in FIS of the Mamdani type can be written as follows [1 to 5]

$$\text{IF } x_1 \text{ is } A_1^{(i)} \text{ AND } x_2 \text{ is } A_2^{(i)} \text{ AND } \dots \text{ AND } x_n \text{ is } A_{pj}^{(i)} \text{ THEN } y \text{ is } B, \quad (1)$$

where $A_1^{(i)}, \dots, A_{pj}^{(i)}$ represent linguistic variables corresponding to the fuzzy sets $\mu_1^{(i)}(x), \mu_2^{(i)}(x), \dots, \mu_{pj}^{(i)}(x), \dots, \mu_m^{(i)}(x), i = 1, \dots, n; j = 1, \dots, m$ and B represents the linguistic variable corresponding to the fuzzy sets $\mu_1(y), \mu_2(y), \dots, \mu_{pk}(y), \dots, \mu_o(y), k = 1, \dots, o$.

The FIS of the Sugeno type can be acquired by modification of FIS of the Mamdani type. These two types of FIS differ in the specification of the output membership functions. These membership functions are constant, linear or polynomial in the case of FIS of the Sugeno type. The division of sets $X_i, i = 1, \dots, n$, into the fuzzy sets $\mu_1^{(i)}(x), \mu_2^{(i)}(x), \dots, \mu_{pj}^{(i)}(x), \dots, \mu_m^{(i)}(x), i = 1, \dots, n; j = 1, \dots, m$ is the same in both types of FISs. Then IR in FIS of the Sugeno type can be written in the following way [2 to 5]

$$\text{IF } x_1 \text{ is } A_1^{(i)} \text{ AND } x_2 \text{ is } A_2^{(i)} \text{ AND } \dots \text{ AND } x_n \text{ is } A_{pj}^{(i)} \text{ THEN } y = h, \quad (2)$$

where h is the constant. In this case the output membership functions are singletons. A FIS containing IRs defined by relation (2) is known as a zero-order FIS of the Sugeno type. If the output membership functions are linear then IRs in FIS of the Sugeno type are in the following form [2 to 5]

$$\text{IF } x_1 \text{ is } A_1^{(i)} \text{ AND } x_2 \text{ is } A_2^{(i)} \text{ AND } \dots \text{ AND } x_n \text{ is } A_{pj}^{(i)} \text{ THEN } y = f(x_1, \dots, x_n), \quad (3)$$

where $f(x_1, \dots, x_n)$ is the linear function. A FIS containing IRs defined by relation (3) is known as a first-order FIS of the Sugeno type. In the case that $f(x_1, \dots, x_n)$ is a polynomial function, it is a second-order FIS of the Sugeno type.

3. Design of Fuzzy Inference System for Prediction of Closing Price of the Index Fund

A FIS to predict the development of the closing price of the index fund (CPIF) can be realised by a passive investment strategy by means of an indexation strategy in the way that CPIF copies the development of the market. Data pre-processing is realised by means of the indicators of technical analysis. The indicators of technical analysis represent the inputs to FIS. The FIS is designed so that the investor is able to predict CPIF in time $t + \Delta t$. A passive investment strategy is suitable for risk-averse investors who plan to achieve the same (or similar) return as the market. In this case the passive investment strategy is realised by means of an indexation strategy. The investor tries to mix the portfolio of securities (the index fund), which copy the returns of the market index. It deals with historical prices of the securities of 35 companies when constructing the index fund (Table 1).

Company	Market capitalisation ¹ [bn. USD]	Industry	The beta factor ²
Alcoa Inc.	31	Metallurgical industry	0.6455
American Express Comp.	53	Financial services	0.0893
AT & T Corp.	86	Telecommunications	0.8733
...
Boeing Comp.	54	Aviation industry	0.7351
BP Amoco Plc.	190	Oil industry	0.4178
Caterpillar Inc.	15	Machinery	0.7932
Cisco Systems Inc.	148	Network equipment	1.7259

Table 1. The basis of companies used for the construction of the index fund

As the given portfolio should copy the development of the market, these companies are chosen on the basis of three criteria. The beta factor of the given security is the first and the most important criterion. It measures the sensitivity of price development of securities to the market movement. The index fund is constructed in such a way that its beta factor is approximately one. In this case the portfolio returns precisely copy the market returns. Selection of securities from different industries is the essence of the second criterion. The portfolio is constructed so that companies from the same industry are not repeated in the portfolio. The significant position of the company in the industry is the third criterion. The position of the company in the industry is measured by its market capitalisation (i.e. number of issued securities multiplied by market price of one security). On the basis of the given criteria 15 companies were selected (they are written in bold font in Table 1). The selection of these companies presents the construction of the indexation strategy. That is, the beta factor of the constructed portfolio approximates to value 1, therefore the portfolio returns copy the market returns. The index can be constructed for observation of the price development of securities involved in the index fund. The index is constructed on the basis of daily data over the course of 4 years. It is a price-weighted index constructed as a simple average. The value 100 points [point] is chosen as the initial value of the index. The opening, highest, lowest and closing prices of securities and volume of the security are

¹ Market capitalisation of companies is from 1. 2. 2001, www.bloomberg.com.

² The beta factor of the given security is calculated on the basis of daily data over the course of 4 years (from 2. 1. 1997 to 29. 12. 2000). The market portfolio return is calculated by means of the S&P 500 index [7].

taken into account when calculating the index. The volume is defined as the number of traded shares on a given day. The development of CPIF is shown in Fig. 1. The closing prices of securities involved in the index fund can be used for calculation of the beta factor of the index fund. As already said, the entire index fund is constructed so that its beta factor is approximately 1. The beta factor of the index fund over the course of the selected period is shown in Fig. 2. It contains the relationship between daily returns of the S&P 500 index (DR_t S&P 500) and daily returns of the index fund (DR_t CPIF). Generally, the returns of the S&P 500 index can be considered as the market returns.

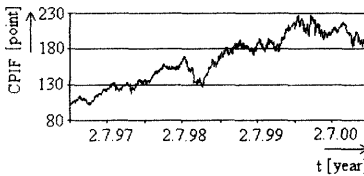


Figure 1. The development of CPIF

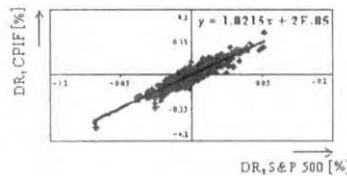


Figure 2. The beta factor of the index fund

The beta factor of the index fund is calculated as a slope of regression line between the daily returns of the S&P 500 index and the daily returns of the index fund. As may be seen from Fig. 2, the beta factor of the index fund is 1.0215. That means, a 1 % change in the returns of the market portfolio (returns of the S&P 500 index) causes a 1.0215 % change in the returns of the index fund. On the basis of this fact, it can be concluded that the constructed index fund copies the market returns in compliance with the conditions of construction of the passive investment strategy. The closing, highest, lowest and opening prices and volume of the index fund can be pre-processed by means of indicators of technical analysis. The calculated indicators can be used as the inputs to FIS. The output of FIS is CPIF in the next week (i.e. $\Delta t = 5$ trading days). Thus indicators of technical analysis in time t (the inputs to FIS) are assigned to CPIF in time $t + \Delta t$ (the output of FIS). Fuzzy inference system appears to be the most suitable with the following input variables: indicator based on the calculation of moving average (MA) – 20 day MA of CPIF, standard deviation (STD) – 20 day STD, momentum (MOM) – 20 day MOM indicator, relative strength index (RSI) – 20 day RSI and William's % R indicator (W%R) – 20 day W%R. The membership functions of these input variables have suitable economic interpretation. The output of FIS is the predicted closing price of the index fund (PCPIF) on 5 trading days. An FIS is shown in Fig. 3. It is a FIS of the Sugeno type with 5 input variables, 3 IRs (which are the result of BIRs extraction from the historical data within the optimisation process of FIS) and 1 output variable. The input variable 20 day MA in time t is represented by means of 3 membership functions. They are Gaussian membership functions. The individual membership functions are described by means of the linguistic variables Low_value_of_MA, Average_value_of_MA, High_value_of_MA. The membership functions are shown in Fig. 4. The other input variables of FIS, i.e. STD, MOM, RSI and W%R are similarly represented. The output variable (PCPIF in time $t + 5$) can be described by means of 3 membership functions. These membership functions are linear, because the FIS is of the Sugeno type. The coefficients of the input and output membership functions of the designed FIS are optimised by the ANFIS method.

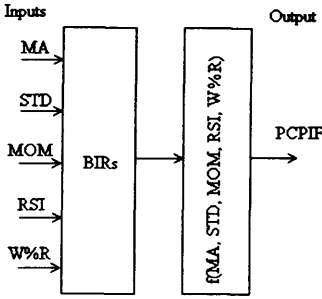


Figure 3. FIS of Sugeno type

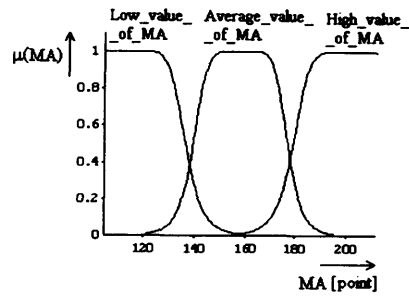


Figure 4. The membership function of MA

The output membership functions can be defined as follows:

$$\mu_1(\text{PCPIF}): \text{PCPIF} = 1.001 * \text{MA} + 0.6824 * \text{STD} + 0.2329 * \text{MOM} - 0.1547 * \text{RSI} + 0.1478 * \text{W\%R} - 9.202, \quad (4)$$

$$\mu_2(\text{PCPIF}): \text{PCPIF} = 1.027 * \text{MA} + 0.496 * \text{STD} + 0.6416 * \text{MOM} - 0.1916 * \text{RSI} + 0.1006 * \text{W\%R} - 56.01, \quad (5)$$

$$\mu_3(\text{PCPIF}): \text{PCPIF} = 1.09 * \text{MA} - 0.4351 * \text{STD} - 0.7775 * \text{MOM} + 0.5834 * \text{RSI} + 0.03815 * \text{W\%R} + 32.95. \quad (6)$$

The BIRs of the designed FIS consist of 3 IRs extracted from the historical data. The individual IRs are weighted by value 1 which means that all IRs have the same influence on the output variable. The IRs of FIS of the Sugeno type are in the following form:

IF (MA is Low_value_of_MA) **AND** (STD is Low_value_of_STD) **AND** (MOM is Average_value_of_MOM) **AND** (RSI is Average_value_of_RSI) **AND** (W%R is High_value_of_W%R) **THEN** (PCPIF is $\mu_1(\text{PCPIF})$), (7)

IF (MA is Average_value_of_MA) **AND** (STD is High_value_of_STD) **AND** (MOM is High_value_of_MOM) **AND** (RSI is High_value_of_RSI) **AND** (W%R is Average_value_of_W%R) **THEN** (PCPIF is $\mu_2(\text{PCPIF})$), (8)

IF (MA is High_value_of_MA) **AND** (STD is Average_value_of_MA) **AND** (MOM is Low_value_of_MOM) **AND** (RSI is Low_value_of_RSI) **AND** (W%R is Low_value_of_W%R) **THEN** (PCPIF is $\mu_3(\text{PCPIF})$). (9)

The designed FIS can be tested on the historical data. The development of CPIF and PCPIF in time $t + \Delta t$ is shown in Fig. 5. The quality of the prediction is measured by means of the root mean square error δ and it reaches the value 5.2673 [%].

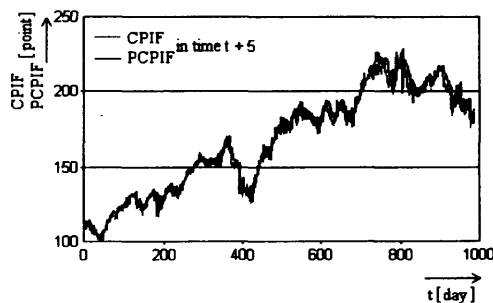


Figure 5. The development of CPIF and PCPIF

4. Conclusion

The paper presents the design of FIS for prediction of CPIF. The passive investment strategy is realised by means of the indexation strategy. The index fund is constructed in such a way that the returns of this fund copy the market portfolio returns. A FIS is designed in a MATLAB environment and is optimised by means of the ANFIS method. The approach of extraction of IRs from the historical data is employed when constructing BIRs of the designed FIS. The error of the prediction of CPIF development allows the application of designed FIS in the real trading system.

References

- [1] Olej, V., Křupka, J. (1996) Analysis of Decision Processes of Automation Control Systems with Uncertainty. [Scientific Monograph], University Press, Elfa, , Košice, Slovak Republic, ISBN 80-88786-29-0, 1-54.
- [2] Kuncheva, L. I. (2000): Fuzzy Classifier Design. A Springer Verlag Company, Germany, ISBN 3-7908-1298-6, 1-313.
- [3] Bandemer, H., Gottwald, S. (1995) Fuzzy Sets, Fuzzy Logic, Fuzzy Methods. John Wiley & Sons, Inc., New York, 9-45.
- [4] Deboeck, G. J. (1994) Trading on the Edge: Neural, Genetic and Fuzzy Systems for Chaotic Financial Markets. John Wiley & Sons, Inc., New York, 189-263.
- [5] Kruse, R., Gebhardt, J., Klawonn, F. (1994) Foundations of Fuzzy Systems. John Wiley & Sons, Inc., New York, 160-185.
- [6] Olej, V. (1999) Evolution Stochastic Optimization Algorithms: Genetic Algorithms and Evolution Strategies. Electrical Engineering Journal, ISSN 1335-3632, Vol.50, No.9-10, 266-272.
- [7] Jilek, J. (1997) Financial markets. Grada Publishing, Praha, ISBN 80-7169-453-3, s.281-307.

Evolutionary Algorithms Designer – Graphical Way of Implementing Evolutionary Algorithms

Peter MACEJ

Dept. of Cybernetics & AI, Technical University of Kosice, Slovakia, Peter.Macej@tuke.sk

Abstract. In this paper the Evolutionary Algorithms Designer (EAD) is presented. EAD provides means to build, maintain and transfer EAs in very intuitive and easy way. It uses graphical interpretation of EA and tries to fill the gap in EA tools area.

Keywords: *evolutionary algorithms, design, graphical tool, Petri Nets, education*

1. Introduction

Seriousness of modern techniques developed in the EA field and the need of their inclusion in different areas implies the need of sophisticated system, which maximally frees the user from routine work, speeds up the design and allows the portability of EA among users. In this paper EAD – Evolutionary Algorithms Designer is introduced. EAD is a graphical tool for creating and running EA, it allows design of EA in a user-friendly way, is universal, free and algorithms created by it are portable.

2. EAD Editor

The main part of EAD is a graphical editor, which provides interaction with the user. It allows creating diagrams representing the EA.

The user defines EA by diagram because graphical representation of a problem is very close to human understanding. Such a model consists of events and their relations. Relations represent time sequence and passing of parameters, in our case populations. Boxes represent processes, mostly genetic operations. In general, every process has inputs and outputs that define data flow and time succession of two processes. Here the data are populations. Since diagrams may be quite complex, modified Petri nets are used for analyzing and running the diagrams.

The whole EA (except of fitness function) is created very user-friendly in graphical environment, so the user can focus on experiment not on its implementation. EAD allows defining very complex and unusual algorithms that would be very hard to implement in a classical way. There is a set of blocks, which the EA consists of, and arrows express relations among them. All parameters are inserted with dialog windows.

It is necessary to say that the tool for inserting fitness function was expressive enough. Lisp language was considered the best because it allows to define or modify code at run time and many researchers from AI area are already familiar with it.

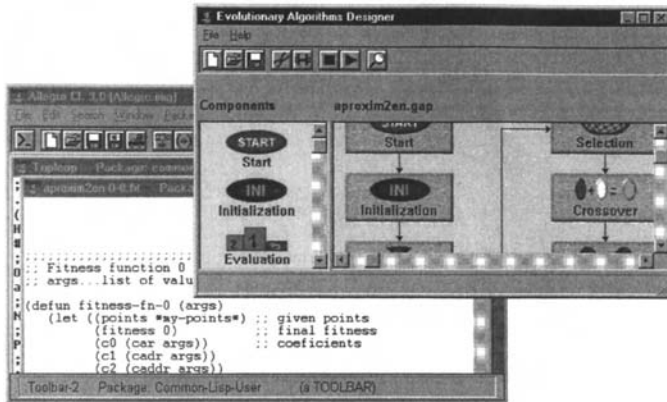


Figure 1. Editor window

3. Conclusion

Evolutionary Algorithms Designer (EAD) provides means to build, maintain and transfer EA in a very intuitive and easy way.

There is a vacuum in the area of graphical editors for EA. ECANSE system from Siemens is such an example, but with some disadvantages. EVOKE is an example of LISP based library, but EA must be defined programmatically **Chyba! Nenašiel sa žiaden zdroj odkazov..**

EAD tries to fill this space by implementing various features and thus it has the ambition to help spreading AE among wider community of users.

References

- [1] EVOKE User's Guide. <ftp://ftp.ifi.ntnu.no/pub/Fag/it378/evoke/evoke.ps>
- [2] Michalewicz, Z.: Genetic Algorithms + Data Structures = Evolution Programs. Springer-Verlag, 1996
- [3] Mitchell, M.: An Introduction to Genetic Algorithms. The MIT Press, 1996

This page intentionally left blank

Author Index

Amari, S.	3	Lampinen, J.	152
Bednár, P.	196	Lažansky, J.	146
Beňušková, L.	17	Liška, M.	286
Bodyanskiy, Y.	29	Macej, P.	342
Boian, E.	251	Mach, M.	279
Buša, J.	40	Maimon, O.	233
Čapkovič, M.	321	Mišík, L.	159
Čerňanský, M.	17	Mokriš, I.	304
Chizi, B.	233	Náplava, P.	135
Chval, J.	221	Navara, M.	111
Cierniak, R.	298	Novotný, D.	61
Cpałka, K.	85	Nowicki, R.	124
Czarnowski, I.	10	Očenášek, J.	227
De Baets, B.	91	Olej, V.	309,336
Dobeš, M.	69	Palko, M.	221
Fedurco, M.	336	Parsopoulos, K.E.	214
Feng, X.	291	Pirč, V.	40
Figedy, Š.	272	Profir, A.	251
Fodor, J.	91	Rutkowski, L.	85,124
Forgáč, R.	304	Šamulka, M.	239,262
Gacôgne, L.	173	Sannoh, A.	255
Gebeová, G.	245	Sato, E.	34
Géczy, P.	3	Scherer, R.	124
Hercek, J.	207	Schwarz, J.	227
Hirota, K.	98,189,329	Šeda, M.	117
Horčík, R.	137	Sekaj, I.	183
Hrehuš, M.	272	Sekanina, L.	166
Hubaľ, M.	196	Shinn-Cunningham, B.	201
Hudec, M.	239,245,262	Shmilovici, A.	233
Iori, M.	255	Sinčák, P.	61
Jakša, R.	48	Šnorek, M.	135
Jędrzejowicz, P.	10	Št'astný, J.	111
Jiřina, M.	24	Strackeljan, J.	291,315
Kaino, T.	329	Takama, Y.	189
Kapustík, I.	207	Tikk, D.	105
Kasabov, N.	141	Tiňo, P.	17
Kolodyazhniy, V.	29	Turčaník, M.	286
Konfršt, Z.	146	Tvrđík, J.	159
Kopčo, N.	201	Tyshchuk, R.V.	130
Kostelník, P.	239,245,262	Usui, S.	3
Kováč, V.	245	Vaščák, J.	98
Kozák, Š.	321	Volná, E.	55
Křivý, I.	159	Vrahatis, M.N.	214
Krokavec, D.	73	Watanabe, S.	34
Kulishova, N.	29	Yamaguchi, T.	34,255

Zadeh, L.A.	83	Zetanova, Z.	279
Zelinski, C.	251		